MASTER EN SCIENCES MATHÉMATIQUES

Analyse de méthodes de calcul de points de départ pour des algorithmes de points intérieurs

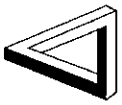
Braibant, Anne-Sophie

Award date:
2003

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

Analyse de méthodes de calcul de points de départ pour des algorithmes de points intérieurs



Mémoire présenté pour l'obtention
du grade de
Licencié en Sciences Mathématiques
par

Anne-Sophie Braibant

Promoteur : **Annick Sartenaer**

Année Académique 2002-2003

Remerciements

Je souhaiterais remercier à travers ces quelques lignes les personnes qui, de près ou de loin, ont contribué à l'élaboration de ce mémoire.

Je tiens à remercier plus particulièrement ma promotrice, Annick Sartenaer, pour ses remarques et encouragements, ainsi que pour sa disponibilité tout au long de cette année, malgré les nouvelles fonctions qui lui ont été attribuées. Ce mémoire fut une expérience très enrichissante tant d'un point de vue formatif que personnel.

Résumé

Le présent travail se situe dans le cadre des méthodes de points intérieurs en programmation linéaire et non linéaire, et concerne plus précisément la comparaison de différentes stratégies de calcul de points de départ pour de telles méthodes.

D'un point de vue théorique, différents algorithmes de points intérieurs de type primal-dual sont présentés pour la résolution de problèmes linéaires et non linéaires. Un algorithme particulier est sélectionné dans le cadre linéaire, algorithme à partir duquel différentes approches de calcul de points de départ seront numériquement mises à l'épreuve et comparées.

D'un point de vue pratique, différentes stratégies de calcul de points de départ proposées dans la littérature pour la programmation linéaire seront tout d'abord testées et comparées. Dans un deuxième temps, une nouvelle heuristique développée dans le cadre des méthodes de points intérieurs pour les problèmes non linéaires sera présentée et particularisée au cadre linéaire mis en place. La nouvelle stratégie ainsi obtenue sera alors testée afin d'évaluer sa pertinence lorsqu'elle est appliquée à des problèmes linéaires, et de comparer sa performance avec les stratégies propres au cas linéaire.

Abstract

This work concerns interior-point methods in linear and nonlinear programming, and more particularly compares several strategies for the computation of starting points for this type of methods.

From a theoretical point-of-view, different primal-dual interior point algorithms are first presented for the solution of linear and nonlinear problems. A specific algorithm is then selected in the linear case and used to test and compare different strategies for the starting point computation.

From a practical point-of-view, several strategies proposed in the literature to compute a starting point in linear programming are tested and compared. A new heuristic is then developed in the frame of interior point methods for nonlinear problems, and adapted to the linear case. This adapted heuristic is then tested in order to evaluate his relevance when applied to linear problems and its performance is compared with the other strategies previously considered.

Table des matières

Introduction

En programmation linéaire tout comme en programmation non linéaire, le calcul de points de départ appropriés pour des algorithmes de points intérieurs est considéré comme une étape importante de la résolution de problèmes. En effet, il est bien connu que les performances de telles méthodes sont influencées par les caractéristiques du point de départ considéré.

L'objectif de ce mémoire est de décrire et analyser différentes approches de calcul de points de départ existantes dans la littérature pour des algorithmes de points intérieurs de type primal-dual en programmation linéaire ainsi qu'en programmation non linéaire. Pour cela, ces différentes stratégies sont appliquées au départ d'une méthode primale-duale de points intérieurs particulière, en programmation linéaire, inspirée de l'algorithme de Mehrotra, puis comparées sur base des résultats obtenus.

Ce mémoire est ainsi constitué de deux parties. Au cours de la première, le procédé général des méthodes primales-duales de points intérieurs en programmation linéaire est présenté, ainsi que quelques algorithmes spécifiques. Trois approches de calcul de points de départ sont ensuite décrites et appliquées au départ d'un de ces algorithmes (l'algorithme MPC), dans le but de résoudre un ensemble de problèmes tests. Cette application permet d'analyser les caractéristiques des points de départ générés et de tester l'efficacité de ces stratégies au départ d'un algorithme particulier.

La deuxième partie commence, quant à elle, par introduire le schéma suivi par une méthode primale-duale de points intérieurs en programmation non linéaire. Une heuristique de calcul de points de départ est ensuite présentée pour des algorithmes de points intérieurs de type primal-dual en programmation non linéaire, puis adaptée au cadre linéaire. Cette heuristique est alors testée au départ de l'algorithme MPC et les résultats obtenus sont analysés et comparés aux résultats obtenus pour les trois approches en programmation linéaire. Les programmes issus de l'implémentation de l'algorithme de points intérieurs choisi et de ces différentes stratégies de calcul de points de départ sont repris en Annexe.

Première partie

Programmation Linéaire

Chapitre 1

Méthodes Primales-Duales de Points Intérieurs en Programmation Linéaire

1.1 Introduction

Pour commencer, il est important de poser correctement le cadre dans lequel nous allons travailler. En programmation linéaire, tout problème traité est en général énoncé sous une forme appelée **forme standard** et définie de la manière suivante :

$$\begin{array}{ll} min & c^T x \\ s.c. : & Ax = b \\ & x \geq 0, \end{array} \tag{1.1}$$

où $c, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times n}$, supposée de rang m. Toutefois, si ce n'est pas le cas, il est toujours possible de se ramener sous cette forme en utilisant, entres autres, des variables d'écart [12, page 22].

Considérant le problème (1.1), le vecteur x de $\mathbb{R}^n$ sera appelé **point réalisable** de celui-ci, s'il satisfait les contraintes $Ax = b$ et $x \geq 0$. Dès lors, il s'en suit que l'ensemble des points réalisables est l'**ensemble réalisable**. Par la suite, nous désignerons par **problème primal**, ou plus communément par **primal**, tout problème de la forme (1.1), et par **variable primale**, toute composante du vecteur x.

A chaque programme primal linéaire peut être associé un autre programme également linéaire. Celui-ci est appelé **problème dual**, ou tout

simplement **dual**, et a la forme :

$$\begin{array}{ll} max & b^T\lambda \\ s.c.: & A^T\lambda + s = c \\ & s \geq 0, \end{array} \tag{1.2}$$

où $\lambda \in \mathbb{R}^m$ et $s \in \mathbb{R}^n$. De la même manière que pour le primal, les composantes du vecteur λ constituent les **variables duales**. Les composantes du vecteur s définissent, quant à elles, les **variables d'écart duales**. Notons aussi que les vecteurs λ et s sont les **multiplicateurs de Lagrange** associés respectivement aux contraintes $Ax = b$ et $x \geq 0$. Par définition des variables d'écart [12, page 22], le problème (1.2) peut se mettre sous une forme plus compacte mais équivalente :

$$\begin{array}{ll} max & b^T\lambda \\ s.c.: & A^T\lambda \leq c, \end{array} \tag{1.3}$$

où $\lambda \in \mathbb{R}^m$. Cependant, nous ne retiendrons que la première formulation du dual, (1.2), pour la suite de notre développement. L'**ensemble réalisable** du programme dual (1.2) est défini par l'ensemble des **points réalisables** (λ, s) qui vérifient les contraintes $A^T\lambda + s = c$ et $s \geq 0$.

L'**ensemble des solutions** du problème primal (1.1) et du problème dual (1.2) sont respectivement donnés par :

$$\Omega_P = \{x^* \mid x^* \text{ résoud } (1.1)\}, \tag{1.4}$$

$$\Omega_D = \{(\lambda^*, s^*) \mid (\lambda^*, s^*) \text{ résoud } (1.2)\}. \tag{1.5}$$

Une théorie de la dualité a été développée, permettant d'établir des relations entre les problèmes (1.1) et (1.2). Ainsi, à partir de l'ensemble réalisable et de l'ensemble des solutions du primal (1.1), des informations sur le dual (1.2) sont obtenues, et vice versa. Par exemple, étant donné un vecteur réalisable x pour (1.1) et un vecteur réalisable (λ, s) pour (1.2), nous avons, par le théorème de dualité faible [12, page 3], que la fonction objectif duale donne une borne inférieure sur la fonction objectif primale, ou encore que la fonction objectif primale constitue une borne supérieure pour la fonction objectif duale, i.e., $b^T\lambda \leq c^Tx$. A la solution, cette inégalité devient une égalité, autrement dit, les fonctions objectifs du primal et du dual coïncident. En effet, si x^* est solution de (1.1) et (λ^*, s^*) est solution de (1.2), le théorème de dualité forte [12, page 3] implique l'égalité $b^T\lambda^* = c^Tx^*$.

Les conditions d'optimalité des problèmes primal et dual sont données par les **conditions nécessaires et suffisantes de Karuch-Kuhn-Tucker**,

dites conditions KKT. Le point (x^*, λ^*, s^*) sera donc solution optimale de la paire de problèmes primale-duale, (1.1) et (1.2), si et seulement si le système :

$$\begin{aligned} A^T\lambda + s &= c, && \text{(1.6a)} \\ Ax &= b, && \text{(1.6b)} \\ x_i s_i &= 0, \ i = 1, ..., n, && \text{(1.6c)} \\ (x, s) &\geq 0, && \text{(1.6d)} \end{aligned}$$

constituant les conditions KKT, est vérifié pour $(x, \lambda, s) = (x^*, \lambda^*, s^*)$. Par conséquent, nous pouvons déterminer **l'ensemble des solutions primales-duales** de la manière suivante :

$$\Omega = \Omega_P \times \Omega_D = \{(x^*, \lambda^*, s^*) \text{ satisfaisant le système (1.6a)-(1.6d)}\}. \quad (1.7)$$

Pour compléter cette introduction, il nous reste à définir le **saut de dualité**. Celui-ci est noté par $x^T s$ et représente la différence entre les fonctions objectifs primale et duale. En effet, par les équations (1.6a) et (1.6b), nous obtenons facilement ce résultat :

$$x^T s = s^T x = (c - A^T\lambda)^T x = c^T x - \lambda^T (Ax) = c^T x - b^T \lambda.$$

Par la relation (1.6d), cette quantité est positive ou nulle. De plus, à la solution, la condition de complémentarité (1.6c) implique que le saut de dualité est nul, i.e., $(x^*)^T s^* = 0$.

1.2 Objectif des méthodes primales-duales de points intérieurs

Après avoir défini le cadre de travail de la programmation linéaire, nous pouvons à présent expliquer le but des méthodes primales-duales de points intérieurs. Ces méthodes consistent à trouver des solutions *primales-duales* (x^*, λ^*, s^*) en appliquant des variantes de la méthode de Newton aux trois conditions d'égalité du système (1.6a)-(1.6d). Pour cela, elles modifient les directions de recherche et les longueurs de pas à chaque itération, afin de vérifier *strictement* la contrainte de positivité (1.6d).

Mathématiquement, il s'agit de résoudre le système non linéaire :

$$F(x, \lambda, s) \equiv \begin{pmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{pmatrix} = 0, \quad (1.8)$$

où $F : \mathbb{R}^{2n+m} \longrightarrow \mathbb{R}^{2n+m}$, $X = diag(x_1, ..., x_n)$, $S = diag(s_1, ..., s_n)$ et $e = (1, ..., 1)^T \in \mathbb{R}^n$, de telle sorte que la solution obtenue vérifie :

$$(x, s) > 0. \tag{1.9}$$

A l'itération k, l'itéré (x^k, λ^k, s^k) est produit de telle manière qu'il se trouve dans l'intérieur strict de l'orthant de non-négativité défini par $(x^k, s^k) \geq 0$, d'où le nom de méthodes de *points intérieurs.*

Pour la plupart de ces méthodes, les itérés sont des points strictement réalisables. Définissant l'**ensemble des points réalisables**, $\mathcal{F}$, et l'**ensemble des points strictement réalisables**, $\mathcal{F}^0$, par :

$$\mathcal{F} = \{(x, \lambda, s) \mid Ax = b,\ A^T\lambda + s = c,\ (x, s) \geq 0\} \tag{1.10}$$

et

$$\mathcal{F}^0 = \{(x, \lambda, s) \mid Ax = b,\ A^T\lambda + s = c,\ (x, s) > 0\}, \tag{1.11}$$

nous avons, à chaque itération, que l'itéré engendré se trouve dans l'ensemble $\mathcal{F}^0$, i.e., $(x^k, \lambda^k, s^k) \in \mathcal{F}^0$. A partir de maintenant, nous supposerons l'ensemble des points strictement réalisables non vide : $\mathcal{F}^0 \neq \emptyset$.

1.3 Concepts de base

1.3.1 Trajectoire centrale

La **trajectoire centrale** est un arc de points strictement réalisables, paramétrisé par un scalaire $\tau > 0$. Elle se note $\mathcal{C}$ et chacun de ses points satisfait le système (1.6a)-(1.6d) modifié, défini par :

$$\begin{aligned} A^T\lambda + s &= c, && \text{(1.12a)} \\ Ax &= b, && \text{(1.12b)} \\ x_i s_i &= \tau,\ i = 1, ..., n, && \text{(1.12c)} \\ (x, s) &> 0. && \text{(1.12d)} \end{aligned}$$

Ce système constitue les conditions KKT modifiées, dites conditions KKT_τ. La trajectoire centrale est donc déterminée par l'ensemble des points, notés $(x_\tau, \lambda_\tau, s_\tau)$, vérifiant le système (1.12a)-(1.12d), pour $\tau > 0$. En reprenant les notations utilisées en (1.8), nous pouvons écrire le système (1.12a)-(1.12d)

sous une autre forme, équivalente, qui est la suivante :

$$F(x_\tau, \lambda_\tau, s_\tau) \equiv \begin{pmatrix} A^T\lambda_\tau + s_\tau - c \\ Ax_\tau - b \\ X_\tau S_\tau e \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \tau e \end{pmatrix}, \qquad (1.13a)$$

$$(x_\tau, s_\tau) > 0, \qquad (1.13b)$$

où $X_\tau = diag((x_\tau)_1, ..., (x_\tau)_n)$ et $S_\tau = diag((s_\tau)_1, ..., (s_\tau)_n)$. Ce système se différencie du système (1.6a)-(1.6d) par le fait que les produits $x_i s_i$ sont non nuls : $x_i s_i = \tau > 0$, $\forall i$. A chaque τ correspond un point différent sur $\mathcal{C}$. Ainsi, considérant la suite décroissante des τ, telle que $\tau_1 > \tau_2 > ... > 0$, la trajectoire centrale nous guide vers une solution primale-duale lorsque τ tend vers zéro. En effet, en faisant tendre τ vers zéro, chaque produit $x_i s_i$ converge également vers zéro, tout en restant strictement positif. Ainsi, le système (1.12a)-(1.12d) approxime de plus en plus le système (1.6a)-(1.6d), dont la solution est une solution primale-duale. Sachant que sur la trajectoire, le point $(x_{\tau_1}, \lambda_{\tau_1}, s_{\tau_1})$ est associé au paramètre τ_1, $\mathcal{C}$ passera successivement par les points $(x_{\tau_1}, \lambda_{\tau_1}, s_{\tau_1})$, $(x_{\tau_2}, \lambda_{\tau_2}, s_{\tau_2})$, ... , pour finalement s'arrêter sur la solution $(x^*, \lambda^*, s^*) \in \Omega$, point correspondant à $\tau = 0$. Toutefois, comme il est assez coûteux de suivre exactement $\mathcal{C}$, les méthodes primales-duales de points intérieurs se contenteront de la suivre approximativement.

1.3.2 Direction de Newton

Avant d'introduire une nouvelle notion, rappelons que notre objectif est de trouver une solution primale-duale du système non linéaire (1.8). Pour y parvenir, nous avons besoin de la **direction de Newton pure**, $(\Delta x, \Delta\lambda, \Delta s)$, dite également **direction affine**. De manière générale, celle-ci est obtenue en résolvant le système linéaire :

$$\nabla F(x, \lambda, s) \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = -F(x, \lambda, s), \qquad (1.14)$$

où F est donné en (1.8). Reprenant cette définition et supposant que le point (x, λ, s) appartient à l'ensemble $\mathcal{F}^0$, les équations de pas (1.14) se développent sous la forme particulière suivante :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe \end{pmatrix}. \qquad (1.15)$$

De par sa définition, la direction de Newton ne tient pas compte des contraintes de positivité. Ainsi, l'itéré (x^k, λ^k, s^k) engendré par un pas complet effectué le long de cette direction se trouve, en général, en dehors de l'orthant de non-négativité. Dans ce cas, il ne satisfait donc pas la contrainte de positivité stricte $(x^k, s^k) > 0$, puisqu'au moins une des composantes du vecteur (x^k, s^k) est négative ou nulle. Pour éviter ce problème, une recherche linéaire est effectuée le long de la direction obtenue. Le nouvel itéré est alors obtenu par la relation :

$$(x, \lambda, s) + \alpha(\Delta x, \Delta\lambda, \Delta s), \tag{1.16}$$

avec $\alpha \in (0, 1]$, tel que $x + \alpha\Delta x > 0$ et $s + \alpha\Delta s > 0$.

Cependant, il n'est souvent possible d'effectuer qu'un petit pas le long de la direction de Newton ($\alpha \ll 1$) avant de violer la contrainte de positivité (1.9). C'est pourquoi les méthodes primales-duales modifient cette direction en une **direction de Newton** dite **modifiée**, orientée *vers l'intérieur* de l'orthant de non-négativité. Pour cela, une fonction barrière est introduite dans le primal (1.1), ainsi que dans le dual (1.2). Ces problèmes deviennent alors respectivement :

$$\begin{array}{ll} min & c^T x - \tau \sum_{i=1}^{n} ln x_i \\ s.c. : & Ax = b, \end{array} \tag{1.17}$$

et

$$\begin{array}{ll} max & b^T \lambda + \tau \sum_{i=1}^{n} ln s_i \\ s.c. : & A^T \lambda + s = c, \end{array} \tag{1.18}$$

où τ est un paramètre de pénalité strictement positif. Le domaine de définition de ces fonctions objectifs est respectivement l'ensemble des points strictement réalisables du primal (1.1) et du dual (1.2). Notant x_τ la solution du problème (1.17) pour chaque valeur $\tau > 0$, x_τ vérifie les conditions d'optimalité suivantes [12, Théorème A.1] :

$$\tau X^{-1} e + A^T \lambda = c, \tag{1.19a}$$

$$Ax = b, \tag{1.19b}$$

$$x > 0, \tag{1.19c}$$

où la matrice $X \in \mathbb{R}^{n \times n}$ est définie à la suite du système (1.8). Si nous définissons les composantes du vecteur s par

$$s_i = \frac{\tau}{x_i}, \qquad i = 1, ..., n,$$

il est évident que les conditions (1.19a)-(1.19c) deviennent identiques aux conditions (1.12a)-(1.12d) définissant la trajectoire centrale. Les valeurs des composantes de x_τ, solution du problème (1.17), correspondent donc aux composantes en x du vecteur $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$. De plus, si le vecteur (λ_τ, s_τ) est considéré comme la solution du problème (1.18), celui-ci constitue en fait les composantes en λ et s du vecteur $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$. Suite à ce raisonnement, la trajectoire centrale peut également être définie comme l'ensemble des points $(x_\tau, \lambda_\tau, s_\tau)$ tels que x_τ est solution de (1.17) et (λ_τ, s_τ) est solution de (1.18). Par conséquent, les algorithmes basés sur les problèmes (1.17) et (1.18) trouvent des approximations des points $(x_\tau, \lambda_\tau, s_\tau)$, pour des valeurs décroissantes de τ. De cette manière, ils suivent la trajectoire centrale vers une solution primale-duale de (1.6a)-(1.6d). La solution de la paire de problèmes (1.17) et (1.18) est donc obtenue en appliquant la méthode de Newton au sytème (1.6a)-(1.6d) modifié, i.e., au système (1.12a)-(1.12d), afin de satisfaire les contraintes de ces problèmes, ainsi que la condition de positivité stricte $(x, s) > 0$. En ce sens, la direction de Newton est dite modifiée. De cette manière, un plus long pas peut être pris le long de la direction calculée avant de sortir de l'orthant de non-négativité. De plus, cette direction empêche les composantes de (x, s) de prendre des valeurs trop proches des valeurs frontalières.

La plupart des algorithmes primals-duals effectuent des pas de Newton modifiés vers les points de $\mathcal{C}$, plutôt que des pas de Newton purs. Pour décrire ces pas de Newton modifiés, nous introduisons deux mesures :

- un *paramètre de centrage* $\sigma \in [0, 1]$,
- une *mesure de dualité* μ, mesurant la valeur moyenne des produits $x_i s_i$ et définie par :

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i s_i = \frac{x^T s}{n}. \tag{1.20}$$

Posant $\tau = \sigma\mu$, la direction de Newton modifiée est dirigée vers un point de $\mathcal{C}$, noté $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu})$. Cette direction se note $(\Delta x, \Delta\lambda, \Delta s)$ et est obtenue en particularisant les équations de pas (1.14) au système (1.13a) pour $\tau = \sigma\mu$, i.e., en résolvant le système :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{pmatrix}. \tag{1.21}$$

Une fois cette direction calculée, les deux objectifs poursuivis par les algorithmes de points intérieurs de type primal-dual sont de *réduire la mesure*

de dualité μ tout en se rapprochant de la trajectoire centrale $\mathcal{C}$, i.e., tout en *améliorant la centralité.* Afin de satisfaire au mieux ces objectifs, la plupart des algorithmes choisissent une valeur de σ dans l'intervalle ouvert $(0,1)$. Pour les valeurs extrémes de σ, il est facile de voir qu'un seul des deux objectifs est rempli. En effet, en posant $\sigma = 1$, les équations de pas (1.21) définissent une *direction de centrage*, orientée vers $(x_\mu, \lambda_\mu, s_\mu) \in \mathcal{C}$, point pour lequel $\tau = \mu$. Celle-ci améliore la centralité, mais réduit de peu, voire de rien, la mesure de dualité μ. Toutefois, le fait de se rapprocher de la trajectoire centrale permettra d'effectuer un plus long pas à la prochaine itération. Par contre, en prenant la valeur $\sigma = 0$, le système (1.21) définit une *direction affine*, qui s'avère être la direction de Newton pure. Elle est orientée vers le point qui représente la solution optimale sur $\mathcal{C}$ et entraîne une réduction de la mesure de dualité μ. Cependant, cette direction n'améliore nullement la centralité.

1.3.3 Cadre primal-dual

Suite aux éléments de base expliqués dans les sections précédentes, nous pouvons résumer le cadre de travail des méthodes primales-duales de points intérieurs par l'algorithme suivant :

Cadre de travail primal-dual
Etant donné $(x^0, \lambda^0, s^0) \in \mathcal{F}^0$;
Pour k=0,1,2,... :

1. résoudre :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -X^k S^k e + \sigma_k \mu_k e \end{pmatrix}, \quad (1.22)$$

   où $\sigma_k \in [0,1]$ et $\mu_k = \frac{(x^k)^T s^k}{n}$;

2. poser $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha_k(\Delta x^k, \Delta \lambda^k, \Delta s^k)$, en choisissant $\alpha_k > 0$ tel que $(x^{k+1}, s^{k+1}) > 0$;

Fin (boucle).

Au cours des sections suivantes, quelques méthodes primales-duales de points intérieurs particulières sont décrites, méthodes basées sur ce cadre de travail primal-dual.

1.4 Algorithmes de suivi de chemin

Les algorithmes de suivi de chemin sont des algorithmes de points intérieurs de type primal-dual dont le cadre de travail correspond au cadre primal-dual [Section 1.3.3]. Avant de décrire en lui-même le procédé de ces algorithmes, quelques propriétés les caractérisant sont à souligner. D'une part, les méthodes de suivi de chemin engendrent des itérés (x^k, λ^k, s^k) *strictement réalisables* et confinés *dans un voisinage de* $\mathcal{C}$, en partant d'un point de départ strictement réalisable. Autrement dit, ces points vérifient les deux premières égalités du système (1.12a)-(1.12d) et la contrainte de positivité stricte (1.12d). D'autre part, au cours des itérations, ces méthodes *suivent approximativement* la trajectoire centrale $\mathcal{C}$, dans la direction faisant diminuer la valeur des composantes du vecteur XSe, i.e., telle que la valeur de τ décroît, pour $\tau = \sigma\mu$ avec $\sigma \in [0, 1]$. En effet, les produits $x_i s_i$ n'ayant généralement pas la même valeur, les itérés produits ne se trouvent pas exactement sur $\mathcal{C}$. La déviation par rapport à $\mathcal{C}$ se mesure en comparant la valeur des produits $x_i s_i$ à celle de la mesure de dualité μ. Elle est donnée par la **mesure de proximité** définie comme suite :

$$\frac{1}{\mu}\|XSe - \mu e\| \quad = \quad \frac{1}{\mu}\left\| \begin{bmatrix} x_1 s_1 \\ \vdots \\ x_n s_n \end{bmatrix} - \left(\frac{x^T s}{n}\right) e \right\|. \tag{1.23}$$

Quelques remarques supplémentaires viennent compléter la définition de cette mesure. Tout d'abord, précisons que la norme-2 comme la norme-∞ sont utilisées pour calculer sa valeur. Pour ces deux normes, si une borne stricte supérieure unité est imposée à la mesure de proximité, il est assuré que la contrainte de positivité stricte (1.12d) est satisfaite [12, page 84] :

$$\frac{1}{\mu}\|XSe - \mu e\| < 1 \quad \Rightarrow \quad (x, s) > 0.$$

De plus, si pour un itéré quelconque (x, λ, s), la mesure de proximité est nulle, cela signifie que ce dernier se trouve sur la trajectoire centrale, i.e. :

$$\frac{1}{\mu}\|XSe - \mu e\| = 0 \quad \Rightarrow \quad (x, \lambda, s) \in \mathcal{C}.$$

Notons que ce résultat est trivial, par définition de la mesure de proximité.

Afin de rester proches de la trajectoire centrale, les algorithmes de suivi de chemin travaillent dans un voisinage de $\mathcal{C}$ tel que la mesure de proximité, relative aux points qui appartiennent à ce voisinage, est petite. Suivant cet objectif, les voisinages utilisés sont :

$$\mathcal{N}_2(\Theta) = \{(x,\lambda,s) \in \mathcal{F}^0 \mid \|XSe - \mu e\|_2 \leq \Theta\mu\}, \text{ où } \Theta \in [0,1)\}, \quad (1.24)$$

$$\mathcal{N}_\infty(\Theta) = \{(x,\lambda,s) \in \mathcal{F}^0 \mid \|XSe - \mu e\|_\infty \leq \Theta\mu\}, \text{ où } \Theta \in [0,1)\}. \quad (1.25)$$

Un troisième voisinage est obtenu si la motivation de ces algorithmes est formulée d'une autre façon : empêcher les produits $x_i s_i$ de prendre une valeur beaucoup plus petite que leur valeur moyenne ou encore les empêcher d'approcher trop vite la frontière de l'orthant de non-négativité. De cette manière, cet autre voisinage est défini par :

$$\mathcal{N}_{-\infty}(\gamma) = \{(x,\lambda,s) \in \mathcal{F}^0 \mid x_i s_i \geq \gamma\mu \ \forall i = 1,...,n, \text{ où } \gamma \in (0,1)\}. \quad (1.26)$$

Pour de petites valeurs de γ, ce voisinage occupe une grande partie de l'ensemble des points strictement réalisables, $\mathcal{F}^0$.

Les algorithmes de suivi de chemin commencent par sélectionner un voisinage parmi $\mathcal{N}_2$, $\mathcal{N}_\infty$ ou $\mathcal{N}_{-\infty}$. Ensuite, une valeur comprise entre $(0,1)$ est attribuée au paramètre de centrage σ et la longueur de pas α est choisie de telle sorte que les itérés restent dans le voisinage sélectionné. Pour rappel, les deux objectifs que doivent satisfaire les algorithmes de points intérieurs de type primal-dual sont de réduire la mesure de dualité μ et d'améliorer la centralité.

Introduisons à présent les notations suivantes, $\forall \alpha \in [0,1]$:

$$(x(\alpha),\lambda(\alpha),s(\alpha)) = (x,\lambda,s) + \alpha(\Delta x, \Delta\lambda, \Delta s), \quad (1.27)$$

$$\mu(\alpha) = \frac{x(\alpha)^T s(\alpha)}{n}. \quad (1.28)$$

Les itérés $(x(\alpha),\lambda(\alpha),s(\alpha))$, engendrés par les algorithmes de suivi de chemin à partir d'un point (x,λ,s) appartenant au voisinage $\mathcal{N}_2(\Theta)$, restent dans ce voisinage pour $\Theta \in (0,1)$, i.e. :

$$(x(\alpha),\lambda(\alpha),s(\alpha)) \in \mathcal{N}_2(\Theta). \quad (1.29)$$

Au cours des itérations, la mesure de dualité définie par (1.28) et relative au nouvel itéré donné par (1.27), converge quant à elle linéairement vers zéro,

à un taux constant d'une valeur $1-\sigma$. La nouvelle valeur de μ est calculée à partir de la relation :

$$\mu(\alpha) = (1 - \alpha(1-\sigma))\mu. \tag{1.30}$$

Notons aussi que la complexité des algorithmes de suivi de chemin est polynomiale. En effet, pour un seuil $\epsilon > 0$ fixé, il existe un indice $K = \mathcal{O}(\sqrt{n}\log\frac{1}{\epsilon})$ tel que la mesure de dualité à l'itération k soit inférieure au seuil ϵ, i.e., $\forall k \geq K$:

$$\mu_k \leq \epsilon. \tag{1.31}$$

Les résultats (1.29), (1.30) et (1.31) sont énoncés de manière générale car ils s'appliquent à tous les algorithmes obtenant leur direction de recherche à partir du système (1.21). Ils sont donc valables en particulier pour les algorithmes de suivi de chemin définis par après. Pour plus de détails, le lecteur peut s'en remettre à la référence [12, Chapitre 5].

Ayant défini les concepts de base propres aux algorithmes de suivi de chemin, les sections suivantes sont consacrées à la description de trois algorithmes particuliers de ce genre : l'algorithme de suivi de chemin à petits pas, dit SPF (« the Short-Step Path-Following Algorithm »), l'algorithme de suivi de chemin prédicteur-correcteur, appelé algorithme PC (« the Predictor-Corrector method ») et enfin l'algorithme de suivi de chemin à grands pas, dit LPF (« the Long-Step Path-Following Algorithm »).

1.4.1 Algorithme à petits pas

L'algorithme SPF commence par prendre $\mathcal{N}_2(\Theta)$ comme voisinage de $\mathcal{C}$ et fixe ensuite les paramètres suivants :

- le paramètre $\Theta = 0.4$,
- la longueur de pas α à 1,
- le paramètre de centrage à $\sigma = 1 - \frac{0.4}{\sqrt{n}}$.

Chosissant comme point de départ un point du voisinage sélectionné, i.e., $(x^0, \lambda^0, s^0) \in \mathcal{N}_2(\Theta)$, tous les itérés engendrés à partir de celui-ci ne quittent pas le voisinage de sorte qu'à chaque itération k :

$$(x^k, \lambda^k, s^k) \in \mathcal{N}_2(0.4).$$

Par la relation (1.30), nous pouvons voir que la mesure de dualité converge linéairemant vers zéro puisque nous avons, pour $k = 0, 1, 2, \ldots$:

$$\mu_{k+1} = \sigma\mu_k = \left(1 - \frac{0.4}{\sqrt{n}}\right)\mu_k.$$

L'algorithme SPF, suivant le cadre de travail primal-dual, est donné par :

Algorithme SPF

Etant donné $\Theta = 0.4$, $\sigma = 1 - \frac{0.4}{\sqrt{n}}$ et $(x^0, \lambda^0, s^0) \in \mathcal{N}_2(\Theta)$;

Pour k=0,1,2,... :

1. poser $\sigma_k = \sigma$ et résoudre (1.22) pour obtenir la direction de recherche $(\Delta x^k, \Delta\lambda^k, \Delta s^k)$;

2. poser $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + (\Delta x^k, \Delta\lambda^k, \Delta s^k)$;

Fin (boucle).

Cet algorithme est illustré à l'aide de la figure 1.1, tirée du livre de S.J. Wright [12, page 87].

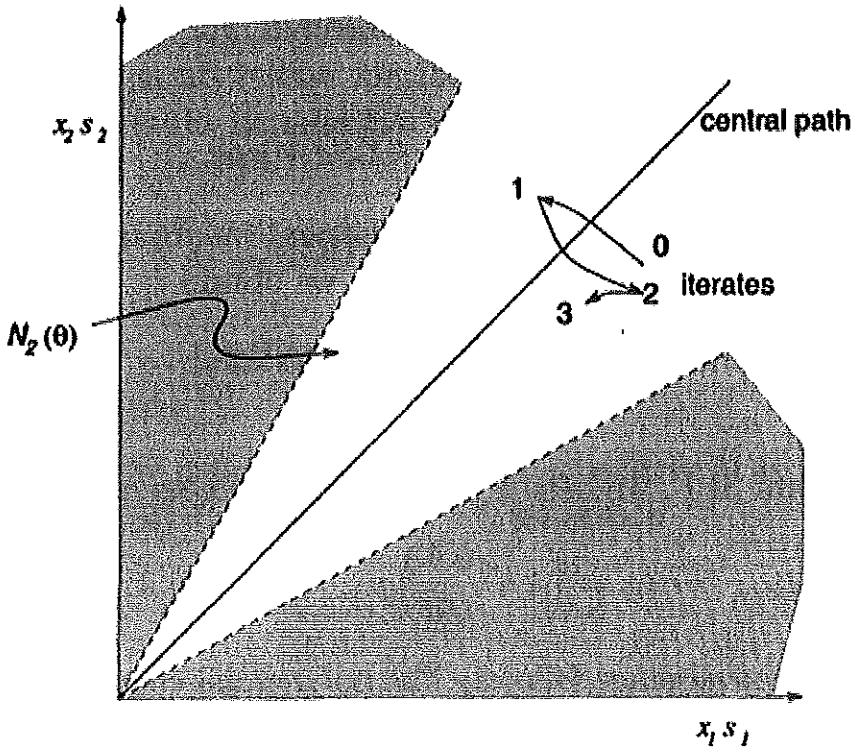


FIG. 1.1 – Itérés de l'Algorithme SPF, dessinés dans l'espace (xs)

Sur cette figure, les premiers itérés, engendrés par l'algorithme SPF pour un problème à deux dimensions, sont repris. Les axes horizontal et vertical représentent respectivement les produits $x_1 s_1$ et $x_2 s_2$, de sorte que la trajectoire centrale est déterminée par la demi-droite partant du point (0,0) et déterminant un angle de $\pi/4$ radians avec l'axe des abscisses. Dans cet espace non linéaire, la direction de recherche est représentée par des courbes plutôt que des lignes. La solution du problème se trouve à l'origine, et l'objectif de cet algorithme est d'atteindre ce point en maintenant les conditions d'égalité $Ax = b$ et $A^T\lambda + s = c$ en tous les itérés.

1.4.2 Algorithme prédicteur-correcteur

L'algorithme prédicteur-correcteur, plus communément appelé algorithme PC, présente deux particularités. Tout d'abord, il sélectionne une *paire de voisinages* emboîtés, du type $\mathcal{N}_2(\Theta)$: $\mathcal{N}_2(0.25) \subseteq \mathcal{N}_2(0.5)$. Ainsi, les itérés pairs se trouvent dans le voisinage interne $\mathcal{N}_2(0.25)$ et les itérés impairs dans le voisinage externe $\mathcal{N}_2(0.5)$, mais pas au-delà. Les valeurs de Θ sont choisies de la sorte pour des raisons de simplicité. Ensuite, au cours des itérations, cet algorithme alterne entre deux sortes de pas :

- des *pas prédicteurs* permettant de réduire la mesure de dualité,
- des *pas correcteurs* permettant d'améliorer la centralité.

Les pas prédicteurs sont calculés aux itérations paires en prenant comme paramètre de centrage $\sigma = 0$. Partant d'un itéré dans $\mathcal{N}_2(0.25)$, la longueur de pas est calculée comme étant la plus grande valeur de α telle que le nouvel itéré se trouve dans $\mathcal{N}_2(0.5)$. La mesure de dualité est ainsi réduite d'un facteur $(1 - \alpha)$, par (1.30), et converge globalement vers zéro au cours des itérations. Notons qu'il existe une borne inférieure $\overline{\alpha}$ sur la longueur de pas α :

$$\overline{\alpha} = min\left(\frac{1}{2}, \left(\frac{\mu}{8\|\Delta X \Delta S\|}\right)^{\frac{1}{2}}\right), \tag{1.32}$$

avec $\Delta X = (\Delta x_1, ..., \Delta x_n)$ et $\Delta S = (\Delta s_1, ..., \Delta s_n)$, permettant de donner une estimation de la valeur dont μ est réduite lors des itérations paires :

$$\mu_{k+1} \leq \left(1 - \frac{0.4}{\sqrt{n}}\right)\mu_k, \; k = 0, 2, 4, ..., \tag{1.33}$$

[12, page 94]. Les pas correcteurs sont, quant à eux, pris aux itérations impaires, en posant $\sigma = 1$ et $\alpha = 1$. Les itérés engendrés par ces pas, à partir d'un point appartenant au voisinage $\mathcal{N}_2(0.5)$, se trouvent dans $\mathcal{N}_2(0.25)$ et

sont donc plus proches de $\mathcal{C}$. Les pas correcteurs n'affectent pas la mesure de dualité. En effet, par (1.30), nous avons la relation $\mu(1) = \mu$. Cependant, une meilleure progression dans la réduction de μ se fera sentir à l'itération suivante.

L'algorithme PC est de la forme suivante :

Algorithme PC

Etant donné $(x^0, \lambda^0, s^0) \in \mathcal{N}_2(0.25)$;

Pour k=0,1,2,... :

- *si k est pair* $\rightarrow$ *pas prédicteur* :

  1. résoudre :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -X^k S^k e \end{pmatrix},$$

  pour obtenir $(\Delta x^k, \Delta \lambda^k, \Delta s^k)$;

  2. choisir α_k = la plus grande valeur de $\alpha \in [0, 1]$ telle que :

$$(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha)) \in \mathcal{N}_2(0.5);$$

  3. poser : $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k(\alpha_k), \lambda^k(\alpha_k), s^k(\alpha_k))$;

  fin (si) ;

- *si k est impair* $\rightarrow$ *pas correcteur* :

  1. résoudre :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -X^k S^k e + \mu_k e \end{pmatrix},$$

  pour obtenir $(\Delta x^k, \Delta \lambda^k, \Delta s^k)$;

2. prendre $\alpha_k = 1$ et poser : $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + (\Delta x^k, \Delta \lambda^k, \Delta s^k)$;

fin (si) ;

Fin (boucle).

Pour illustrer les deux premières itérations de cet algorithme, nous avons repris la figure 1.2 suivante, [12, page 93].

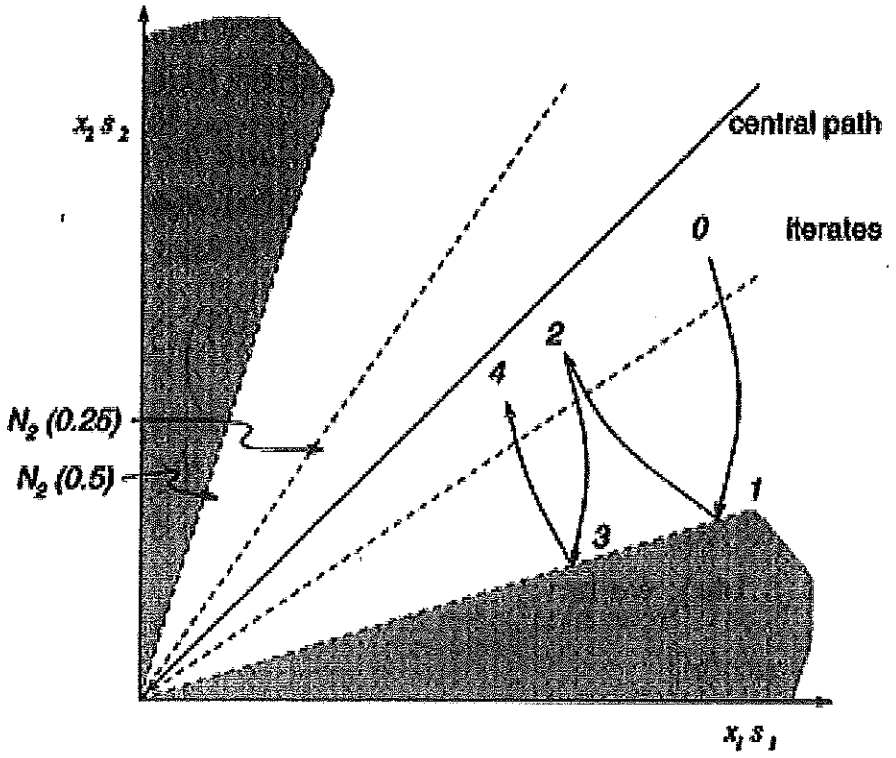


FIG. 1.2 – Itérés de l'Algorithme PC, dessinés dans l'espace (xs)

Partant d'un point (x^0, λ^0, s^0) dans le voisinage interne $\mathcal{N}_2(0.25)$, un pas prédicteur est calculé en posant $\sigma_0 = 0$. On se déplace alors le long de cette direction, jusqu'à la frontière du voisinage externe $\mathcal{N}_2(0.5)$, afin de déterminer le nouvel itéré (x^1, λ^1, s^1). Ensuite, un pas correcteur est déterminé en posant $\sigma_1 = 1$. Un pas unité est effectué le long de la direction obtenue, menant ainsi à l'itéré suivant (x^2, λ^2, s^2), qui se trouve à l'intérieur du voisinage

interne. Ces deux étapes sont ainsi répétées, générant une suite d'itérés se trouvant en alternance dans le voisinage interne, puis externe.

1.4.3 Algorithme à grands pas

L'algorithme LPF génère des itérés dans le voisinage $\mathcal{N}_{-\infty}(\gamma)$, où, par exemple, $\gamma = 10^{-3}$. Contrairement aux deux algorithmes précédents, celui-ci ne fixe pas une valeur uniforme pour le paramètre de centrage σ. En effet, à chaque itération, il choisit comme paramètre σ, une valeur située dans l'intervalle $[\sigma_{min}, \sigma_{max}]$, dont les bornes fixées sont telles que $0 < \sigma_{min} < \sigma_{max} < 1$. Une fois la direction de recherche obtenue en résolvant le système (1.21), la longueur de pas est choisie pour être la plus grande valeur de α telle que le nouvel itéré soit dans $\mathcal{N}_{-\infty}(\gamma)$. En ce qui concerne la mesure de dualité, celle-ci décroît à chaque itération selon l'inégalité suivante :

$$\mu_{k+1} \leq \left(1 - \frac{\delta}{n}\right) \mu_k, \quad \forall k \geq 0, \tag{1.34}$$

où δ est une constante indépendante de n, [12, Théorème 5.11].

Dès lors, l'algorithme formel est décrit par :

Algorithme LPF

Etant donné $\gamma, \sigma_{min}, \sigma_{max}$ avec $\gamma \in (0,1)$, $0 < \sigma_{min} < \sigma_{max} < 1$, et $(x^0, \lambda^0, s^0) \in \mathcal{N}_{-\infty}(\gamma)$;

Pour k=0,1,2,... :

1. choisir $\sigma_k \in [\sigma_{min}, \sigma_{max}]$;
2. résoudre le système (1.22) pour obtenir $(\Delta x^k, \Delta\lambda^k, \Delta s^k)$;
3. choisir α_k = la plus grande valeur de $\alpha \in [0,1]$ telle que :
$$(x(\alpha), \lambda(\alpha), s(\alpha)) \in \mathcal{N}_{-\infty}(\gamma);$$
4. poser $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k(\alpha_k), \lambda^k(\alpha_k), s^k(\alpha_k))$;

Fin (boucle).

Un comportement typique de l'algorithme LPF est décrit par la figure 1.3 ci-dessous, extraite du livre de S.J. Wright [12, page 97].

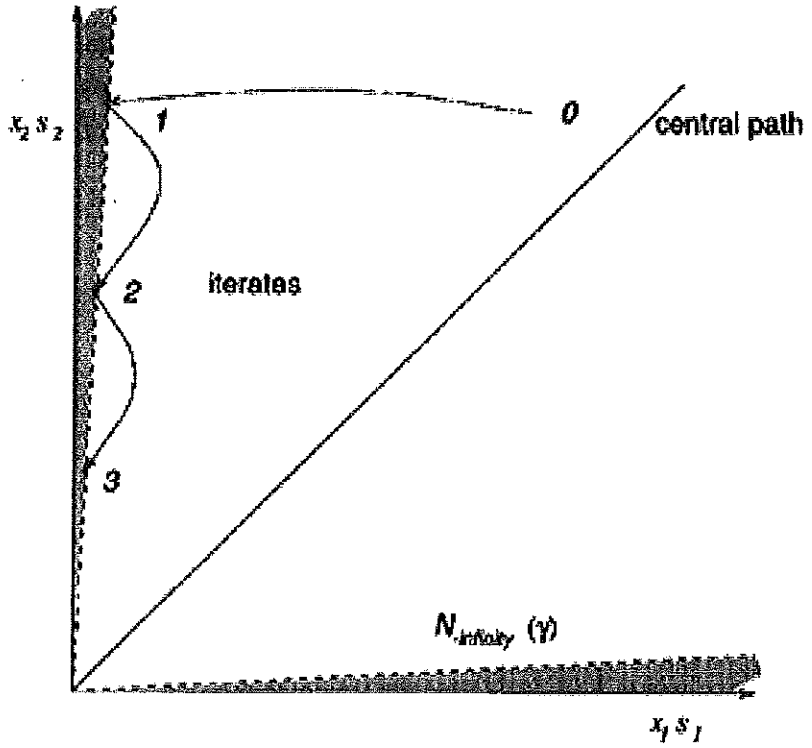


FIG. 1.3 – Itérés de l'Algorithme LPF, dessinés dans l'espace (xs)

1.4.4 Convergence de la suite des itérés

Pour compléter cette section, analysons le comportement limite de la suite des itérés $\{(x^k, \lambda^k, s^k)\}$. Pour cela, une suite de points $\{(x^k, \lambda^k, s^k)\}$, générée par un des algorithmes de suivi de chemin décrits précédemment, est prise en compte. Si la mesure de dualité μ_k, relative à ces itérés, décroît vers zéro, lorsque $k \to \infty$, nous avons que la suite $\{(x^k, s^k)\}$ est bornée et admet donc au moins un point limite [12, Théorème 5.14]. Dans ce cas, il reste à s'assurer que le point limite de la suite $\{(x^k, \lambda^k, s^k)\}$ est une solution primale-duale. Ce résultat s'obtient simplement en sachant qu'il est possible de construire une solution primale-duale à partir de n'importe quel point limite de $\{(x^k, s^k)\}$, en utilisant le raisonnement suivant. Considérons $\mathcal{K}$, la sous-suite pour laquelle $\lim_{k \in \mathcal{K}}(x^k, s^k) = (x^*, s^*)$. Nous avons alors, $\forall k \in \mathcal{K}$, les relations :

$$Ax^k = b, \quad c - s^k \in Im(A^T), \quad (x^k, s^k) > 0.$$

En passant à la limite, le point (x^*, s^*) satisfait :

$$Ax^* = b, \quad c - s^* \in Im(A^T), \quad (x^*, s^*) \geq 0, \quad (x^*)^T s^* = 0,$$

sachant que l'ensemble $Im(A^T)$ est fermé et que $\mu_k \downarrow 0$. Dès lors, l'égalité $c - s^* = A^T \lambda^*$ est vérifiée pour un certain λ^*. En comparant ces relations d'égalité et d'inégalité au point limite avec les conditions du système (1.6a)-(1.6d), il s'en suit que :

$$(x^*, \lambda^*, s^*) \in \Omega,$$

i.e., la limite de la suite des itérés $\{(x^k, \lambda^k, s^k)\}$ est une solution primale-duale.

Ce résultat de convergence clôture la description des algorithmes de suivi de chemin. Dans la section suivante, une autre méthode primale-duale de points intérieurs est abordée, à savoir l'algorithme de points intérieurs non-réalisables.

1.5 Algorithme de points intérieurs non-réalisables

Dans le cadre des algorithmes de suivi de chemin, le point de départ était considéré comme un point strictement réalisable. Malheureusement, il n'est pas toujours possible de trouver un tel point. Une manière de contourner ce problème est d'utiliser une méthode n'exigeant pas que le point de départ satisfasse cette propriété. Il existe d'autres possibilités permettant de faire face à ce problème que nous ne développerons pas ici [12, page 107]. Dans cette section, un algorithme de points intérieurs non-réalisables est décrit, l'algorithme IPF (« Infeasible-Interior-Point Algorithm »). Cet algorithme est en fait une variante de l'algorithme LPF [Section 1.4.3], pour laquelle le point de départ (x^0, λ^0, s^0) est un point non-réalisable. Ce point n'appartient donc pas à l'ensemble $\mathcal{F}^0$ défini en (1.11). La seule condition qui lui est imposée est de satisfaire strictement la contrainte de positivité :

$$(x^0, \lambda^0, s^0) \text{ tel que } (x^0, s^0) > 0. \tag{1.35}$$

Cet algorithme est donc valable lorsque $\mathcal{F}^0 = \emptyset$. Les itérés engendrés à partir d'un tel point de départ sont également des points non-réalisables, bien que leurs points limites soient des points optimaux admissibles.

Avant de présenter l'algorithme décrivant le cadre de travail de cette méthode de points intérieurs, introduisons une nouvelle notion, celle de résidu. Le **résidu** mesure la quantité par laquelle les deux premières contraintes

d'égalité du système (1.6a)-(1.6d) sont violées. Pour chacune de ces égalités, cette quantité est respectivement définie par le vecteur suivant :

$$r_b = Ax - b, \tag{1.36}$$
$$r_c = A^T\lambda + s - c, \tag{1.37}$$

où $r_b \in \mathbb{R}^m$ et $r_c \in \mathbb{R}^n$. Evalués au point $(x, \lambda, s) = (x^k, \lambda^k, s^k)$, ces vecteurs sont notés r_b^k et r_c^k.

L'algorithme suivant, dont les différentes étapes sont commentées par la suite, résume le schéma suivi par la méthode de points intérieurs non-réalisables.

Algorithme IPF

Etant donné $\gamma, \beta, \sigma_{min}, \sigma_{max}$ avec $\gamma \in (0,1), \beta \geq 1$,
et $0 < \sigma_{min} < \sigma_{max} \leq 0.5$;

Choisir $(x^0, \lambda^0, s^0) \in \mathcal{N}_{-\infty}(\gamma, \beta)$;

Pour k=0,1,2,... :

1. choisir $\sigma_k \in [\sigma_{min}, \sigma_{max}]$ et résoudre :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} -r_c^k \\ -r_b^k \\ -X^k S^k e + \sigma_k \mu_k e \end{pmatrix}; \tag{1.38}$$

2. choisir α_k = la plus grande valeur de $\alpha \in [0,1]$ telle que :

$$(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha)) \in \mathcal{N}_{-\infty}(\gamma, \beta)$$

et que la condition d'Armijo suivante tient :

$$\mu_k(\alpha) \leq (1 - 0.1\alpha)\mu_k, \tag{1.39}$$

où $(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha))$ et $\mu_k(\alpha)$ sont définis en (1.28) ;

3. poser $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k(\alpha_k), \lambda^k(\alpha_k), s^k(\alpha_k))$;

Fin (boucle).

Etant basé sur l'algorithme LPF, l'algorithme IPF sélectionne un voisinage qui est défini comme une extension du voisinage $\mathcal{N}_\infty(\theta)$, décrit en (1.25), admettant des points non-réalisables :

$$\mathcal{N}_{-\infty}(\gamma,\beta) = \{(x,\lambda,s) \mid \|(r_b,r_c)\| \leq \beta\frac{\|(r_b^0,r_c^0)\|}{\mu_0}\mu,\ (x,s) > 0, \qquad (1.40)$$
$$x_i s_i \geq \gamma\mu, i = 1,...,n\},$$

où

- $\gamma \in (0,1)$ et $\beta \geq 1$ sont des paramètres donnés,
- (r_b^0, r_c^0) et μ_0 donnent respectivement la valeur des résidus et de la mesure de dualité évalués au point de départ (x^0, λ^0, s^0).

La condition "$\beta \geq 1$" assure que le point de départ (x^0, λ^0, s^0) se trouve dans $\mathcal{N}_{-\infty}(\gamma,\beta)$. L'algorithme se poursuit en résolvant les équations de Newton modifiées, données par le système (1.21), où les définitions (1.36) et (1.37) sont introduites dans le membre de droite, i.e., en résolvant le système (1.38). Deux nouvelles conditions sont alors imposées sur la longueur de pas α, déterminée le long de la direction ainsi obtenue, afin que le nouvel itéré reste dans le voisinage $\mathcal{N}_{-\infty}(\gamma,\beta)$. Tout d'abord, ce pas est calculé de sorte que la quantité par laquelle les contraintes $A^T\lambda + s = c$ et $Ax = b$ sont violées décroît au moins aussi rapidement que la mesure de dualité μ. Autrement dit, la première condition assure que le nouvel itéré satisfait l'inégalité :

$$\|(r_b,r_c)\| \leq \beta\frac{\|(r_b^0,r_c^0)\|}{\mu_0}\mu.$$

La deuxième condition imposée sur α assure que μ diminue d'au moins une petite fraction de la décroissance prédite à chaque itération. Cette condition, appelée condition d'Armijo, est donnée par la relation (1.39).

Faisant tendre μ_k monotonement vers zéro et gardant tous les itérés (x^k, λ^k, s^k) dans le voisinage $\mathcal{N}_{-\infty}(\gamma,\beta)$, l'algorithme IPF assure que les résidus correspondants convergent vers zéro, i.e., $r_b^k \to 0$ et $r_c^k \to 0$, pour $k \to \infty$. De cette manière, la valeur des résidus relatifs à la solution primale-duale (x^*, λ^*, s^*) est nulle : $r_b^* = 0$ et $r_c^* = 0$. Par conséquent, le point (x^*, λ^*, s^*) vérifie les contraintes d'égalité et est un point admissible.

Terminons la description de cet algorithme en donnant quelques résultats concernant sa convergence et sa complexité. D'une part, nous avons que la suite $\{\mu_k\}$, générée au cours des itérations de l'algorithme, converge $Q-$linéairement vers zéro. De même, la suite composée des normes des résidus engendrés à chaque itération, $\{\|(r_b^k, r_c^k)\|\}$, converge $\mathcal{R}-$linéairement vers zéro [12, Théorème 6.1]. D'autre part, la complexité de cet algorithme

est de type polynomiale [12, page 112] : il est possible de trouver un K de l'ordre de $n^2|\log\epsilon|$ pour un $\epsilon > 0$, i.e., $K = \mathcal{O}\left(n^2|\log\epsilon|\right)$, de sorte que tous les itérés $\{(x^k, \lambda^k, s^k)\}$ engendrés par l'algorithme IPF satisfassent l'inégalité $\mu_k \leq \epsilon$, $\forall k \geq K$ [12, Théorème 6.2].

Clôturant ce chapitre, la section suivante décrit une cinquième et dernière méthode primale-duale de points intérieurs en programmation linéaire, vue dans le cadre de ce mémoire. Cette méthode consiste en l'algorithme Prédicteur-Correcteur de Mehrotra et se base sur l'algorithme PC [Section 1.4.2].

1.6 Algorithme de Mehrotra (MPC)

Partant d'un point de départ non-admissible, l'algorithme de Mehrotra génère une suite d'itérés (x^k, λ^k, s^k) strictement positifs en leurs composantes x et s, i.e., $(x^k, s^k) > 0$. Il se différencie de l'algorithme PC par le fait que la direction de recherche, calculée à chaque itération pour trouver le nouvel itéré, est constituée de trois composantes : une composante *affine "prédicteur"*, une composante *de centrage* et une composante *correctrice*. En fait, la direction de Newton modifiée de base, donnée par (1.21), est ajustée par une composante correctrice. Précisons que cette correction ne rend pas pour autant les calculs plus coûteux. De plus, cette méthode se distingue aussi de l'algorithme PC par le fait qu'elle effectue des pas différents le long de la composante primale et de la composante duale de la direction de recherche.

Afin de décrire les concepts de base de l'approche de Mehrotra, une variante particulière est donnée par l'algorithme MPC (« Mehrotra Predictor-Corrector Algorithm ») suivant.

Algorithme MPC

Etant donné (x^0, λ^0, s^0) tel que $(x^0, s^0) > 0$;

Pour k=0,1,2,... :

1. résoudre le système suivant pour trouver la direction affine "prédicteur" :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^{aff} \\ \Delta\lambda^{aff} \\ \Delta s^{aff} \end{pmatrix} = \begin{pmatrix} -r_c^k \\ -r_b^k \\ -X^k S^k e \end{pmatrix}, \quad (1.41)$$

où : $r_b^k = Ax^k - b$ et $r_c^k = A^T\lambda^k + s^k - c$;

2. calculer :

$$\begin{aligned} \alpha_{aff}^{pri} &= \arg\max\{\alpha \in [0,1] \mid x^k + \alpha\Delta x^{aff} \geq 0\}, \\ \alpha_{aff}^{dual} &= \arg\max\{\alpha \in [0,1] \mid s^k + \alpha\Delta s^{aff} \geq 0\}, \\ \mu_{aff} &= (x^k + \alpha_{aff}^{pri}\Delta x^{aff})^T(s^k + \alpha_{aff}^{dual}\Delta s^{aff})/n; \end{aligned}$$

3. poser $\sigma = (\mu_{aff}/\mu)^3$;

4. résoudre :

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^{cc} \\ \Delta\lambda^{cc} \\ \Delta s^{cc} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\Delta X_{aff}^k \Delta S_{aff}^k e + \sigma_k \mu_k e \end{pmatrix}, \tag{1.42}$$

où :

$$\begin{aligned} \Delta X_{aff} &= diag(\Delta x_1^{aff}, ..., \Delta x_n^{aff}), \\ \Delta S_{aff} &= diag(\Delta s_1^{aff}, ..., \Delta s_n^{aff}); \end{aligned}$$

5. calculer la direction de recherche et les longueurs de pas jusqu'à la frontière de l'orthant de non-négativité :

$$(\Delta x^k, \Delta\lambda^k, \Delta s^k) = (\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff}) + (\Delta x^{cc}, \Delta\lambda^{cc}, \Delta s^{cc}),$$

$$\begin{aligned} \alpha_{max}^{pri} &= \arg\max\{\alpha \geq 0 \mid x^k + \alpha\Delta x^k \geq 0\}, \\ \alpha_{max}^{dual} &= \arg\max\{\alpha \geq 0 \mid s^k + \alpha\Delta s^k \geq 0\}; \end{aligned}$$

6. poser $\alpha_k^{pri} = \min(0.995 * \alpha_{max}^{pri}, 1)$ et $\alpha_k^{dual} = \min(0.995 * \alpha_{max}^{dual}, 1)$;

7. poser

$$\begin{aligned} x^{k+1} &= x^k + \alpha_k^{pri}\Delta x^k, \\ (\lambda^{k+1}, s^{k+1}) &= (\lambda^k, s^k) + \alpha_k^{dual}(\Delta\lambda^k, \Delta s^k); \end{aligned}$$

Fin (boucle).

A chaque itération, les différentes étapes suivies par cet algorithme dans le but de calculer la direction de recherche pour déterminer le nouvel itéré sont les suivantes. Tout d'abord, la première composante de cette direction, appelée direction affine "prédicteur" et notée $(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$, est calculée en résolvant le système (1.38) pour $\sigma = 0$, i.e., en résolvant (1.41). Une fois cette direction déterminée, la longueur du pas α_{aff}^{pri} est évaluée. Ce pas mesure la distance pouvant être effectuée le long de la composante primale Δx^{aff}, tout en respectant la contrainte de non-négativité, i.e., de sorte que $x^k + \alpha\Delta x^{aff} \geq 0$. De manière similaire, la longueur de pas α_{aff}^{dual} est déterminée. La mesure de dualité est alors évaluée le long de la direction affine, dans le but de calculer la valeur du paramètre de centrage σ, intervenant dans le calcul de la deuxième composante de la direction de recherche. Nous reviendrons par la suite sur ce point.

L'algorithme se poursuit ensuite en résolvant le système (1.42), système identique à celui résolu pour déterminer la composante affine, à savoir (1.41), le membre de droite les différenciant. Cette étape a pour objectif de calculer la direction $(\Delta x^{cc}, \Delta\lambda^{cc}, \Delta s^{cc})$, qui résulte du calcul combiné des composantes de centrage et correctice (cc) de la direction de recherche. En effet, la direction de centrage $(\Delta x^c, \Delta\lambda^c, \Delta s^c)$ est obtenue en résolvant le système (1.38) où le membre de droite prend la forme $(0, 0, \sigma\mu e)^T$, tandis que la matrice des coefficients du membre de gauche est inchangée. La direction correctrice $(\Delta x^{cor}, \Delta\lambda^{cor}, \Delta s^{cor})$ est quant à elle calculée en remplaçant le membre de droite du système (1.38) par $(0, 0, -\Delta X_{aff}\Delta S_{aff})$, où les matrices ΔX_{aff} et ΔS_{aff} sont définies à la suite du système (1.42). Soulignons que la composante correctrice a pour but de compenser la non-linéarité observée lorsqu'un pas complet est effectué le long de la direction affine "prédicteur". Elle modifie la direction de recherche de telle manière que les produits $x_i s_i$ deviennent de plus en plus proches de leur valeur objectif zéro.

Finalement, la direction de recherche totale $(\Delta x, \Delta\lambda, \Delta s)$ est déterminée en additionnant les trois directions calculées, composante par composante. De nouveau, les pas effectués respectivement le long de la composante primale Δx et le long de la composante duale $(\Delta\lambda, \Delta s)$ sont calculés séparément de sorte que le nouvel itéré se trouve dans l'orthant de non-négativité défini par $(x, s) \geq 0$.

L'avantage de calculer la direction affine indépendamment et avant la direction combinée de centrage et correctrice est de pouvoir choisir de manière adaptée la valeur du paramètre de centrage σ, en fonction de l'efficacité de la direction affine. Pour cela, la mesure de dualité est évaluée au point engendré par un pas complet le long de cette direction affine, jusqu'à la frontière de l'orthant de non-négativité. Cette mesure définie au cours de la deuxième

étape de l'algorithme MPC est reprise ci-dessous :

$$\mu_{aff} = \frac{(x + \alpha_{aff}^{pri} \Delta x^{aff})^T (s + \alpha_{aff}^{dual} \Delta s^{aff})}{n}. \tag{1.43}$$

Deux cas de figure peuvent alors se présenter :

1. $\mu_{aff} \ll \mu$: la direction affine est une bonne direction de recherche. En effet, elle entraîne une grande réduction de la valeur de μ. Un léger centrage est donc nécessaire et une valeur proche de zéro est affectée à σ.
2. $\mu_{aff} < \mu$: la direction affine n'est pas une bonne direction de recherche. Une petite réduction est observée dans la valeur de μ. Un centrage plus important est effectué, permettant de mieux progresser à l'itération suivante. Ainsi, le paramètre σ prend une valeur proche de l'unité.

Par conséquent, Mehrotra propose l'heuristique suivante pour le calcul du paramètre de centrage :

$$\sigma = \left(\frac{\mu_{aff}}{\mu}\right)^3. \tag{1.44}$$

Il est évident que cette heuristique est efficace, affectant une valeur adaptée à σ. En effet, dans le cas où nous avons $\mu_{aff} \ll \mu$, σ est proche de zéro, tandis que dans le cas où une petite différence est observée entre les valeurs de μ_{aff} et μ, σ est proche de 1.

Cette heuristique clôture le premier chapitre dont l'objectif était de décrire les méthodes primales-duales de points intérieurs en programmation linéaire, dans un premier temps de manière générale, puis de manière plus particulière. Nous allons à présent nous consacrer au calcul du point de départ pour ces méthodes. Le chapitre suivant reprend donc quelques méthodes de calcul de points de départ. Celles-ci sont tout d'abord expliquées une à une, puis comparées d'un point de vue théorique.

Chapitre 2

Calcul de Points de Départ en Programmation Linéaire

2.1 Introduction

Dans le premier chapitre, nous avons vu que la convergence de l'algorithme IPF est assurée à partir d'un point de départ (x^0, λ^0, s^0) se trouvant à l'intérieur de l'orthant de positivité défini par $(x, s) \geq 0$ [Section 1.5]. La plupart des logiciels étant basés sur cet algorithme, une grande flexibilité apparaît dans le choix du point de départ. Toutefois, selon le mathématicien S.J. Wright [12, page 224], un bon point de départ doit satisfaire deux conditions supplémentaires :

1. Ce point doit être *bien centré*, de telle sorte que les produits $x_i s_i$ sont similaires pour $i = 1, ..., n$. Autrement dit, le point de départ doit être proche de la trajectoire centrale $\mathcal{C}$.
2. Le caractère de *non-admissibilité* relatif à ce point doit être *petit*. Ce caractère se mesure en comparant la norme du vecteur des résidus et la mesure de dualité au point de départ :

$$\frac{\|(r_b^0, r_c^0)\|}{\mu_0}. \tag{2.1}$$

Plus ce rapport est petit, meilleur est le point de départ. Rappelons que la notion de résidu est définie au premier chapitre par les relations (1.36) et (1.37).

Au cours des sections suivantes, trois approches différentes permettant de calculer le point de départ des méthodes primales-duales de points intérieurs

en programmation linéaire sont décrites. Rappelons que ce type de méthode permet de trouver une solution optimale à la paire de problèmes primale-duale (1.1) et (1.2). Ces problèmes sont respectivement repris ci-dessous :

$$\begin{aligned} min \quad & c^T x \\ s.c. : \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{2.2}$$

où $c, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times n}$, de rang m,

$$\begin{aligned} max \quad & b^T \lambda \\ s.c. : \quad & A^T \lambda + s = c \\ & s \geq 0, \end{aligned} \tag{2.3}$$

où $\lambda \in \mathbb{R}^m$ et $s \in \mathbb{R}^n$. La description de ces trois approches de calcul de point de départ est basée sur les formulations (2.2) et (2.3) des problèmes primal et dual.

Remarque : Par la suite, le domaine de définition des vecteurs et matrice de données, c, b et A, est inchangé. Dès lors, il ne sera pas rappelé à chaque fois que nous utilisons ces données. Seul le domaine de définition des inconnues, à savoir x, λ et s, sera indiqué quand nous le jugerons nécessaire.

2.2 Approche de Mehrotra

Afin de trouver un point de départ (x^0, λ^0, s^0) pour les algorithmes de points intérieurs de type primal-dual, S. Mehrotra proposa une heuristique dont les différentes étapes sont décrites ci-dessous. Celle-ci permet de générer un point intérieur, autrement dit, un point satisfaisant strictement les contraintes de positivité de (2.2) et (2.3). De plus, ce point s'avère être bien centré par rapport à la trajectoire centrale et la valeur du rapport (2.1) s'y rapportant est petite. Ce point se trouve donc près de la trajectoire centrale, $\mathcal{C}$, et la valeur des résidus lui correspondants, à savoir r_b^0 et r_c^0, est petite.

La première étape de l'approche proposée consiste à résoudre deux problèmes quadratiques, dont la solution est à la base du calcul du point de départ (x^0, λ^0, s^0) :

$$\begin{aligned} min_x \quad & \|x\|_2^2 \\ s.c. : \quad & Ax = b, \end{aligned} \tag{2.4}$$

où $x \in \mathbb{R}^n$, et

$$\begin{aligned} min_{(\lambda,s)} \quad & \|s\|_2^2 \\ s.c. : \quad & A^T\lambda + s = c, \end{aligned} \tag{2.5}$$

où $\lambda \in \mathbb{R}^m$ et $s \in \mathbb{R}^n$. Leur solution, notée $(\tilde{x}, \tilde{\lambda}, \tilde{s})$, est telle que les vecteurs $\tilde{x}$ et $\tilde{s}$ sont des vecteurs de norme minimale (en norme 2) pour lesquels les résidus r_b et r_c sont nuls, et est donnée par les relations suivantes [9] :

$$\tilde{x} = A^T(AA^T)^{-1}b, \tag{2.6a}$$

$$\tilde{\lambda} = (AA^T)^{-1}Ac, \tag{2.6b}$$

$$\tilde{s} = c - A^T\tilde{\lambda}, \tag{2.6c}$$

où $A^T(AA^T)^{-1}$ est un inverse à droite de A, appelé *inverse généralisé de Penrose* [11]. Au cours de la seconde étape, une série de calculs est effectuée dans le but d'obtenir l'inégalité stricte $(x^0, s^0) > 0$. En effet, le point $(\tilde{x}, \tilde{\lambda}, \tilde{s})$, obtenu par les égalités (2.6a)-(2.6c), est un point vérifiant les contraintes d'égalité des problèmes (2.2) et (2.3). Ses composantes en x et s n'étant pas nécessairement strictement positives, il ne peut être considéré comme le point de départ (x^0, λ^0, s^0). Dès lors, une certaine valeur est ajoutée aux composantes des vecteurs $\tilde{x}$ et $\tilde{s}$ pour obtenir un point strictement positif appartenant à l'orthant de positivité défini par $(x, s) > 0$. Pour déterminer la valeur à ajouter, deux scalaires, δ_x et δ_s, sont tout d'abord évalués :

$$\delta_x = max(-1.5 * min\{\tilde{x}_i\}, 0), \tag{2.7}$$

$$\delta_s = max(-1.5 * min\{\tilde{s}_i\}, 0). \tag{2.8}$$

A partir de ceux-ci, deux autres réels, $\tilde{\delta_x}$ et $\tilde{\delta_s}$, sont calculés de la manière suivante :

$$\tilde{\delta_x} = \delta_x + 0.5 * \frac{(\tilde{x} + \delta_x e)^T(\tilde{s} + \delta_s e)}{\sum_{i=1}^{n}(\tilde{s}_i + \delta_s)}, \tag{2.9}$$

$$\tilde{\delta_s} = \delta_s + 0.5 * \frac{(\tilde{x} + \delta_x e)^T(\tilde{s} + \delta_s e)}{\sum_{i=1}^{n}(\tilde{x}_i + \delta_x)}, \tag{2.10}$$

et placés respectivement dans les vecteurs ϕ et ϕ' de $\mathbb{R}^n$:

$$\phi = \tilde{\delta_x} e \quad \text{et} \quad \phi' = \tilde{\delta_s} e. \tag{2.11}$$

La dernière étape de cette approche permet d'obtenir le point de départ recherché en utilisant la relation :

$$(x^0, \lambda^0, s^0) = (\tilde{x} + \phi, \tilde{\lambda}, \tilde{s} + \phi'). \tag{2.12}$$

Concluons cette partie en montrant que le point (x^0, λ^0, s^0), calculé à partir de (2.12), se trouve à l'intérieur de l'orthant de non-négativité défini par $(x, s) \geq 0$. Pour cela, remarquons tout d'abord par les définitions (2.7) et (2.8) que δ_x et δ_s sont des réels positifs ou nuls :

$$\delta_x \geq 0 \quad \text{et} \quad \delta_s \geq 0. \tag{2.13}$$

De plus, il suit de ces mêmes définitions que les vecteurs $\tilde{x} + \delta_x e$ et $\tilde{s} + \delta_s e$ sont strictement positifs en chacune de leurs composantes, respectivement lorsque $\delta_x > 0$ et $\delta_s > 0$:

$$\tilde{x}_i + \delta_x > 0 \quad \text{et} \quad \tilde{s}_i + \delta_s > 0, \tag{2.14}$$

$i = 1, ..., n$. A partir de la relation (2.13), quatre situations peuvent se présenter à l'issue du calcul de δ_x et δ_s.

1. $\delta_x > 0$ et $\delta_s > 0$: Par définition, cela signifie que les vecteurs $\tilde{x}$ et $\tilde{s}$ possèdent au moins une composante strictement négative. D'une part, il suit de (2.14) que les réels $\tilde{\delta_x}$ et $\tilde{\delta_s}$, donnés par (2.9) et (2.10) vérifient les inégalités strictes : $\tilde{\delta_x} > 0$ et $\tilde{\delta_s} > 0$. D'autre part, nous avons également que les valeurs de $\tilde{\delta_x}$ et de $\tilde{\delta_s}$ sont respectivement plus grandes que la valeur absolue de chaque composante négative de $\tilde{x}$ et $\tilde{s}$. En effet, par (2.7) et (2.8), nous avons les relations :

   $$\delta_x > |\tilde{x}_i| \quad \text{et} \quad \delta_s > |\tilde{s}_j|, \tag{2.15}$$

   $\forall i$ tel que $\tilde{x}_i < 0$ et $\forall j$ tel que $\tilde{s}_j < 0$, avec $i, j \in \{1, ..., n\}$. Ajoutant alors un nombre strictement positif à δ_x, resp. δ_s, il suit de (2.15) que :

   $$\tilde{\delta_x} > |\tilde{x}_i| \quad \text{et} \quad \tilde{\delta_s} > |\tilde{s}_j|, \tag{2.16}$$

   $\forall i$ tel que $\tilde{x}_i < 0$ et $\forall j$ tel que $\tilde{s}_j < 0$, avec $i, j \in \{1, ..., n\}$. Dès lors, le fait que le point de départ soit strictement positif en ses composantes x et s se déduit facilement :

   $$x^0 = \tilde{x} + \tilde{\delta_x} e > 0 \quad \text{et} \quad s^0 = \tilde{s} + \tilde{\delta_s} e > 0. \tag{2.17}$$

2. $\delta_x > 0$ et $\delta_s = 0$: Dans ce cas, au moins une composante de $\tilde{x}$ est négative puisque $\delta_x > 0$ et le fait que δ_s soit nul se traduit en disant que le vecteur $\tilde{s}$ est un vecteur positif ou nul. Supposons ici que $\tilde{s}$ n'est pas le vecteur nul, i.e., $\exists\, i \in \{1, ..., n\}$ (au moins un) tel que :

   $$\tilde{s}_i > 0. \tag{2.18}$$

Par conséquent, le réel $\tilde{\delta_s}$, obtenu de la manière suivante :

$$\tilde{\delta_s} = 0.5 * \frac{(\tilde{x} + \delta_x e)^T \tilde{s}}{\sum_{i=1}^{n} (\tilde{x}_i + \delta_x)}, \tag{2.19}$$

est strictement positif au vu de (2.14) et (2.18). Il est alors évident que le vecteur s^0, donné en (2.12), est strictement positif :

$$s^0 > 0. \tag{2.20}$$

Partant d'une valeur strictement positive pour δ_x, le fait que $x^0 = \tilde{x} + \delta_x e$ soit strictement positif a été prouvé dans le premier cas. On peut donc conclure ce deuxième cas par la relation :

$$x^0 > 0. \tag{2.21}$$

Remarque : Pour le cas où $\tilde{s}$ est le vecteur nul, le lecteur peut s'en remettre à la référence [9].

3. $\delta_x = 0$ et $\delta_s > 0$: Dans ce cas, au moins une composante de $\tilde{s}$ est négative, tandis que $\tilde{x}$ est un vecteur postif ou nul. Limitons-nous au cas où $\tilde{x}$ n'est pas le vecteur nul, i.e., $\exists\, i \in \{1, ..., n\}$ (au moins un) tel que :

$$\tilde{x}_i > 0. \tag{2.22}$$

Dès lors, il découle de (2.14) et (2.22) que la valeur de $\tilde{\delta_x}$, calculée par la relation :

$$\tilde{\delta_x} = 0.5 * \frac{\tilde{x}^T (\tilde{s} + \delta_s e)}{\sum_{i=1}^{n} (\tilde{s}_i + \delta_s)}, \tag{2.23}$$

est strictement positive. Par conséquent, le vecteur $x^0 = \tilde{x} + \delta_x e$ est bien strictement positif :

$$x^0 > 0. \tag{2.24}$$

Notons que le fait que $s^0 = \tilde{s} + \delta_s e$ est strictement positif a déjà été montré lors du premier cas :

$$s^0 > 0. \tag{2.25}$$

Remarque : Plus de détails concernant le cas où $\tilde{x}$ est le vecteur nul sont donnés dans [9].

4. $\delta_x = 0$ et $\delta_s = 0$: Ce dernier cas correspond à un cas particulier pour lequel les vecteurs $\tilde{x}$ et $\tilde{s}$ sont respectivement des points réalisables du primal (2.2) et du dual (2.3). En effet, $\tilde{x}$ vérifie la contrainte d'égalité du primal, de par sa définition, ainsi que la contrainte de non-négativité par le fait que δ_x est nul. Le même raisonnement appliqué à $\tilde{s}$, pour le dual, montre que $\tilde{s}$ en est un point réalisable. De plus, le saut de dualité, donné par $\tilde{x}^T\tilde{s}$, est nul. En effet, cette valeur s'obtient en utilisant les relations (2.6a)-(2.6c) :

$$\begin{aligned}\tilde{x}^T\tilde{s} &= b^T(AA^T)^{-T}A(c - A^T(AA^T)^{-1}Ac) \\ &= b^T(AA^T)^{-T}Ac - b^T(AA^T)^{-T}AA^T(AA^T)^{-1}Ac \\ &= b^T(AA^T)^{-T}Ac - b^T(AA^T)^{-T}Ac \\ &= 0.\end{aligned}$$

Par conséquent, le point $(\tilde{x}, \tilde{\lambda}, \tilde{s})$ est une solution primale-duale optimale de la paire de problèmes (2.2) et (2.3).

Nous avons ainsi décrit chaque situation que nous pouvions rencontrer à partir du calcul de δ_x et de δ_s, et prouvé que le point de départ, obtenu à l'issue de la troisième étape de cette approche, s'avère dans chaque cas satisfaire strictement la contrainte de positivité, exception faite du dernier cas.

2.3 Approche de Gondzio, Andersen, Mészáros et Xu

La méthode développée par J. Gondzio et ses confrères est une variante de celle de S. Mehrotra décrite dans le paragraphe précédent. Elle suit donc le même schéma de construction pour obtenir le point de départ. Une fois calculé, ce point s'avère de nouveau être strictement positif. De plus, il vérifie deux autres conditions, à savoir qu'il est bien centré et que la valeur de (2.1) lui étant relative est petite. Toutefois, cette approche se différencie de la première par quelques éléments sur lesquels nous nous arrêterons dans un prochain paragraphe.

Décrivons à présent le procédé de la méthode. Dans un premier temps, deux problèmes quadratiques sont résolus, pour un paramètre ρ prédéterminé ($\rho = 1$, dans les tests effectués au Chapitre 3). Ces problèmes sont les suivants :

$$\begin{aligned} min \quad & c^T x + \tfrac{\rho}{2}\|x\|_2^2 \\ s.c. : \quad & Ax = b, \end{aligned} \tag{2.26}$$

où $x \in \mathbb{R}^n$, et

$$\begin{array}{ll} min & b^T\lambda + \frac{\rho}{2}(\|\lambda\|_2^2 + \|s\|_2^2) \\ s.c. : & A^T\lambda + s = c \\ & \lambda = 0, \end{array} \tag{2.27}$$

où $\lambda \in \mathbb{R}^m$ et $s \in \mathbb{R}^n$. La solution de cette paire de problèmes est notée $(\tilde{x}, \tilde{\lambda}, \tilde{s})$ et est à la base du calcul du point de départ recherché. Dans un deuxième temps, le procédé se poursuit en construisant deux vecteurs de $\mathbb{R}^n$, ϕ et ϕ', dont les composantes satisfont les relations suivantes, pour $i = 1, ..., n$:

$$\phi_i = \begin{cases} \delta & \text{si } \tilde{x}_i < \delta, \\ 0 & \text{sinon,} \end{cases} \tag{2.28}$$

$$\phi'_i = \begin{cases} \delta & \text{si } \tilde{s}_i < \delta, \\ 0 & \text{sinon,} \end{cases} \tag{2.29}$$

où δ est un paramètre choisi de telle sorte qu'il existe une constante c strictement positive satisfaisant les inégalités strictes : $\delta > c > 0$. Par exemple, $\delta = 1$ convient. Cette valeur sera utilisée pour les tests numériques qui suivront. Finalement, les composantes du point $(\tilde{x}, \tilde{s})$ sont modifiées en conséquence, afin de satisfaire la contrainte de positivité stricte $(\tilde{x}, \tilde{s}) > 0$. Les composantes de $(\tilde{x}, \tilde{s})$ dont la valeur est strictement plus petite que le paramètre δ sont remplacées par celui-ci, les autres restant inchangées :

$$\tilde{x}_i = \begin{cases} \phi_i & \text{si } \tilde{x}_i < \delta, \\ \tilde{x}_i & \text{sinon,} \end{cases} \tag{2.30}$$

$$\tilde{s}_i = \begin{cases} \phi'_i & \text{si } \tilde{s}_i < \delta, \\ \tilde{s}_i & \text{sinon,} \end{cases} \tag{2.31}$$

pour $i = 1, ..., n$. Le point de départ est alors donné par le nouveau point $(\tilde{x}, \tilde{\lambda}, \tilde{s})$ dont les composantes en x et s sont définies par (2.30) et (2.31) :

$$(x^0, \lambda^0, s^0) = (\tilde{x}, \tilde{\lambda}, \tilde{s}). \tag{2.32}$$

Cette méthode engendre bien un point de départ strictement positif puisque toutes les composantes de celui-ci valent au minimum δ, un réel strictement positif.

2.4 Approche de Zhang

La méthode de Y. Zhang est assez simple et s'applique à de nombreux problèmes, sous forme standard, comme les problèmes de programmation

linéaire, les problèmes quadratiques ou encore les problèmes de complémentarité linéaire.

Cette approche permet de calculer un point de départ des algorithmes de points intérieurs de type primal-dual, en résolvant un système de la forme :

$$Mx + Ns - h = 0, \tag{2.33}$$

où $x, s, h \in \mathbb{R}^n$ et $M, N \in \mathbb{R}^{n\times n}$. Notre objectif étant de trouver un point de départ pour les méthodes de résolution des problèmes linéaires (2.2) et (2.3), particularisons le système (2.33) sous la forme adéquate. Pour cela, il est utile de rappeler les deux premières équations du système (1.6a)-(1.6d) vu au premier chapitre :

$$Ax - b = 0, \tag{2.34a}$$
$$A^T\lambda + s - c = 0, \tag{2.34b}$$

où $x, s \in \mathbb{R}^n$ et $\lambda \in \mathbb{R}^m$. Le système (2.34a)-(2.34b) est transformé en un système du type (2.33) en multipliant l'équation (2.34b) par une certaine matrice B. Cette dernière est une matrice de $\mathbb{R}^{(n-m)\times n}$ satisfaisant les conditions suivantes :

- B est de rang ligne plein $n - m$,
- $BA^T = 0$, i.e., les lignes de B forment une base du noyau de la matrice A.

Dès lors, le système (2.34a)-(2.34b) prend la forme suivante :

$$\begin{pmatrix} A \\ 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ B \end{pmatrix} s - \begin{pmatrix} b \\ Bc \end{pmatrix} = 0, \tag{2.35}$$

où $\begin{pmatrix} A \\ 0 \end{pmatrix} \in \mathbb{R}^{n\times n}$, $\begin{pmatrix} 0 \\ B \end{pmatrix} \in \mathbb{R}^{n\times n}$ et $\begin{pmatrix} b \\ Bc \end{pmatrix} \in \mathbb{R}^n$, qui est un système équivalent au système (2.33). La méthode de Zhang se poursuit donc en résolvant (2.35), système dont la solution est désignée par le vecteur $(\tilde{x}, \tilde{s})$. Ensuite, connaissant la valeur de $\tilde{s}$, la méthode calcule le vecteur λ en résolvant l'équation (2.34b) :

$$A^T\lambda + \tilde{s} = c, \tag{2.36}$$

où $\lambda \in \mathbb{R}^m$. La valeur de λ ainsi obtenue est notée par $\tilde{\lambda}$. Finalement, le procédé se termine en choisissant le point de départ (x^0, λ^0, s^0) de telle sorte qu'il satisfasse, composante par composante, la contrainte de positivité stricte :

$$(x^0, s^0) > 0, \tag{2.37}$$

et que le point $(\tilde{x}, \tilde{s})$ soit une borne inférieure de (x^0, s^0). Cette dernière condition est notée :

$$(x^0, s^0) \geq (\tilde{x}, \tilde{s}), \tag{2.38}$$

et signifie que $x_i^0 \geq \tilde{x}_i$ et $s_i^0 \geq \tilde{s}_i$, pour $i = 1, ..., n$. Par exemple, posant $\zeta \in \mathbb{R}^n$ un vecteur strictement positif en toutes ses composantes, une possibilité est de choisir (x^0, λ^0, s^0) de la manière suivante :

$$\begin{aligned} x_i^0 &= max(\zeta_i, \tilde{x}_i), \ i = 1, ..., n, \\ \lambda^0 &= \tilde{\lambda}, \\ s_i^0 &= max(\zeta_i, \tilde{s}_i), \ i = 1, ..., n, . \end{aligned} \tag{2.39}$$

Il est facile de voir que ce point se trouve bien à l'intérieur de l'orthant de non-négativité défini par $(x, s) \geq 0$. Notons que lors des tests numériques effectués au chapitre suivant, le vecteur ζ est fixé au vecteur unité.

2.5 Comparaison théorique entre les différentes approches

Cette partie est consacrée à la comparaison théorique des trois méthodes décrites au cours des sections précédentes, reprenant ainsi les points communs, de même que ceux qui les différencient.

Chacune de ces approches permet de générer un point de départ *strictement positif* pour les méthodes primales-duales de points intérieurs résolvant (2.2) et (2.3). Pour cela, elles calculent toutes trois au préalable un point *vérifiant les contraintes d'égalité* des problèmes (2.2) et (2.3). Ce point est noté $(\tilde{x}, \tilde{\lambda}, \tilde{s})$ et permet de générer le point de départ (x^0, λ^0, s^0). Remarquons toutefois que pour chacune des méthodes, $(\tilde{x}, \tilde{\lambda}, \tilde{s})$ n'est pas un point réalisable des problèmes (2.2) et (2.3). En effet, ce dernier ne satisfait pas les contraintes de positivité, $x \geq 0$ et $s \geq 0$, et peut donc posséder des composantes négatives en x et s.

Comme il est indiqué dans le paragraphe précédent, le point (x^0, λ^0, s^0) est engendré par $(\tilde{x}, \tilde{\lambda}, \tilde{s})$. Quelle que soit la méthode sélectionnée, la composante en λ du point de départ prend la *valeur du point* $\tilde{\lambda}$, aucune contrainte de positivité n'étant imposée sur λ. Par contre, en ce qui concerne les composantes en x et s de (x^0, λ^0, s^0), les trois approches se différencient dans la manière de les calculer. Selon la méthode de Mehrotra, les composantes du vecteur $\tilde{x}$, resp. $\tilde{s}$, sont modifiées en ajoutant à ce dernier un vecteur ϕ donné en (2.11), resp. ϕ', également défini en (2.11). *Chaque* composante de $\tilde{x}$, resp.

$\tilde{s}$, est donc augmentée de la valeur $\tilde{\delta_x}$ calculée par (2.9), resp. $\tilde{\delta_s}$ donnée par (2.10). Notons que, de par sa définition, la valeur dont les composantes de $\tilde{x}$ et $\tilde{s}$ sont modifiées, dépend de ces composantes en question. De cette façon, le point strictement positif (x^0, s^0) est obtenu :

$$(x^0, s^0) = (\tilde{x} + \phi, \tilde{s} + \phi').$$

Remarquons que dans le cas où les scalaires δ_x et δ_s, définis en (2.7) et (2.8), valent tous deux zéro, *aucune* des composantes de $\tilde{x}$ et $\tilde{s}$ n'est modifiée, puisque le point $(\tilde{x}, \tilde{\lambda}, \tilde{s})$ est une solution optimale des problèmes (2.2) et (2.3) [Section 2.2] :

$$(x^*, \lambda^*, s^*) = (\tilde{x}, \tilde{\lambda}, \tilde{s}).$$

Pour Gondzio et ses confrères, les composantes de $\tilde{x}$ et $\tilde{s}$ sont, dans un premier temps, comparées à un réel strictement positif noté δ. Suivant le procédé décrit à la Section 2.3, *certaines* de leurs composantes sont ensuite remplacées par cette valeur δ, les autres étant inchangées :

$$(x^0, s^0) = (\tilde{x}, \tilde{s}),$$

où $\tilde{x}$ et $\tilde{s}$ sont respectivement donnés par les relations (2.30) et (2.31), et sont des points strictement positifs. Toutefois, dans le cas particulier où $\tilde{x}$ et $\tilde{s}$ sont des vecteurs strictement négatifs, i.e., tels que $\tilde{x}_i < 0$ et $\tilde{s}_i < 0$, pour $i = 1, ..., n$, *toutes* leurs composantes sont modifiées afin d'obtenir un point strictement positif :

$$(x^0, s^0) = (\delta e, \delta e),$$

avec e le vecteur unité dans $\mathbb{R}^n$, tandis que si toutes les composantes de $\tilde{x}$, resp. $\tilde{s}$, sont strictement supérieures à δ, le point de départ est donné par :

$$(x^0, s^0) = (\tilde{x}, \tilde{s}),$$

où $\tilde{x}$ et $\tilde{s}$ sont les composantes en x et en s de la solution de la paire de problèmes (2.26)-(2.27).

Finalement, l'approche de Zhang décrit une troisième manière de procéder, similaire à celle utilisée par l'approche de Gondzio. Si le vecteur $(\tilde{x}, \tilde{s})$ est plus grand ou égal au vecteur (ζ, ζ), composante par composante, ce point est considéré comme le point de départ :

$$(\tilde{x}, \tilde{s}) \geq (\zeta, \zeta) \Rightarrow (x^0, s^0) = (\tilde{x}, \tilde{s}),$$

avec ζ un vecteur strictement positif dans $\mathbb{R}^n$. Par contre, si au moins une composante de $\tilde{x}$ et/ou de $\tilde{s}$ est strictement plus petite que la composante correspondante de ζ, cette composante est remplacée par la composante relative de ζ, les autres composantes étant inchangées. Par conséquent, le point (x^0, s^0) est obtenu en modifiant les composantes de $(\tilde{x}, \tilde{s})$ de la manière suivante :

$$\forall\, i \text{ tel que } \tilde{x}_i < \zeta_i \Rightarrow x_i^0 = \zeta_i,$$
$$\forall\, j \text{ tel que } \tilde{s}_j < \zeta_j \Rightarrow s_j^0 = \zeta_j,$$

avec $i, j \in \{1, ..., n\}$. Soulignons que si toutes les composantes de $\tilde{x}$, resp. $\tilde{s}$, sont strictement inférieures aux composantes respectives de ζ, le point de départ est donné par le vecteur $(x^0, s^0) = (\zeta, \zeta)$. Notons que les composantes de ζ sont indépendantes de la valeur de $(\tilde{x}, \tilde{s})$. Le point (x^0, s^0) est donc strictement positif. Dès lors, nous pouvons conclure que *soit certaines* composantes de $\tilde{x}$ et/ou $\tilde{s}$ sont modifiées (voire *toutes*), *soit aucune* de leurs composantes ne l'est.

Pour terminer cette comparaison théorique, le tableau de la page suivante reprend les points principaux de comparaison sur lesquels nous nous sommes arrêtés au cours de cette section, et cela pour chacune des approches décrites dans ce chapitre.

Approche de	Mehrotra	Gondzio et ses confrères	Zhang
$(\tilde{x}, \tilde{\lambda}, \tilde{s})$	vérifie les contraintes d'égalité de (2.2) et (2.3).		
La composante λ^0 du point de départ est donnée par :	$\tilde{\lambda}$	$\tilde{\lambda}$	$\tilde{\lambda}$
Afin d'obtenir (x^0, s^0), les composantes modifiées de $(\tilde{x}, \tilde{s})$ sont :	toutes les composantes, voire aucune (cas particulier).	certaines composantes, voire toutes ou aucune (cas particulier).	certaines composantes, voire toutes ou aucune (cas particulier).
(x^0, λ^0, s^0) est un point	strictement positif en x et s.		

TAB. 2.1 – Points principaux de comparaison des méthodes de calcul de points de départ, pour des algorithmes de points intérieurs de type primal-dual en programmation linéaire.

Le chapitre suivant est consacré à l'implémentation des trois méthodes décrites dans cette partie. Ces méthodes sont utilisées afin de calculer un point de départ pour l'algorithme MPC de Mehrotra [Section 1.6] et des tests numériques sont effectués sur une gamme de problèmes dans le but de les comparer.

Chapitre 3

Résultats numériques

3.1 Introduction

Afin de mettre en pratique la théorie vue au cours des chapitres précédents, nous avons choisi d'implémenter l'algorithme de Mehrotra, dit algorithme MPC [Section 1.6], et de lui appliquer, tour à tour, les trois méthodes de calcul du point de départ vues au chapitre précédent. Rappelons que l'utilisation de cet algorithme a pour but de fournir une solution primale-duale d'un problème primal du type :

$$\begin{array}{ll} min & c^T x \\ s.c. : & Ax = b \\ & x \geq 0, \end{array} \tag{3.1}$$

où $c, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times n}$, de rang m. L'implémentation de ces algorithmes a été effectuée à l'aide du logiciel mathématique Matlab (« MATrix LABoratory ») et les programmes résultant sont repris en Annexe.

Dès lors, l'objectif de ce chapitre est d'effectuer une comparaison entre les trois méthodes de calcul du point de départ, appliquées à l'algorithme MPC, de manière à tenter de répondre aux questions telles que : « L'algorithme MPC résout-il un problème du type (3.1) plus rapidement si une approche est utilisée plutôt qu'une autre ? Le point de départ obtenu est-il meilleur lors de l'utilisation d'une approche plutôt que d'une autre ?, ... ». Pour cela, l'agorithme MPC est utilisé dans le but de résoudre une gamme de problèmes tests, créés de manière aléatoire. Au cours de la section suivante, quelques explications supplémentaires sont fournies quant à l'implémentation des algorithmes et la création de ces problèmes.

3.2 Détails d'implémentation

En ce qui concerne l'algorithme MPC, le schéma décrivant ce dernier [Section 1.6] a été suivi lors de l'implémentation. La condition d'arrêt utilisée s'applique à la valeur de la mesure de dualité μ ainsi qu'à celle de la norme des résidus r_b et r_c. En effet, nous avons vu au premier chapitre que l'objectif des algorithmes de points intérieurs de type primal-dual était de trouver une solution primale-duale d'un problème de type (3.1), satisfaisant les conditions d'optimalité (1.6a)-(1.6d), i.e., une solution telle que $\mu = 0$, $r_b = 0$ et $r_c = 0$. Dès lors, fixant un paramètre ϵ à 10^{-5}, l'algorithme MPC s'arrête lorsqu'un des itérés engendrés (x^k, λ^k, s^k) satisfait au mieux ces relations, i.e., lorsque les valeurs de μ, $||r_b||_2$ et $||r_c||_2$ qui lui sont relatives sont telles que :

$$\mu < 10^{-5},\ ||r_b||_2 < 10^{-5} \text{ et } ||r_c||_2 < 10^{-5}. \tag{3.2}$$

Notons qu'un nombre maximal d'itérations est fixé. Celui-ci est choisi de sorte qu'il prend la valeur du nombre de variables du problème à résoudre, à savoir n, variant ainsi en fonction de ce nombre.

D'autre part, nous avons vu au cours du chapitre précédent que les méthodes de Gondzio et ses confrères, et de Zhang nécessitaient l'utilisation de paramètres, dont les valeurs ont été fixées précédemment. Soulignons que ces valeurs sont suggérées par les différents auteurs lors de la présentation de ces méthodes, [1] et [13]. Pour plus de clarté, nous avons choisi de rappeler la valeur attribuée à chaque paramètre à l'aide du tableau suivant, ainsi que le rôle de chacun dans la méthode de calcul de points de départ correspondante.

paramètre	valeur attribuée	rôle
méthode de Gondzio et ses confrères		
$\rho \in \mathbb{R}$	1	intervient dans les fonctions objectifs des problèmes (2.26) et (2.27).
$\delta \in \mathbb{R}$	1	intervient dans la détermination des points x^0 et s^0.
méthode de Zhang		
$\zeta \in \mathbb{R}^n$	$e \in \mathbb{R}^n$	intervient dans la détermination des points x^0 et s^0.

TAB. 3.1 – Valeurs attribuées aux paramètres utilisés par les méthodes de Gondzio et ses confrères, et de Zhang.

Décrivons à présent la manière dont les problèmes tests de la forme (3.1) sont générés. Tout d'abord, fixant le nombre de variables n, ainsi

que le nombre de contraintes d'égalité m, le vecteur $c \in \mathbb{R}^n$ et la matrice $A \in \mathbb{R}^{m \times n}$, de rang m, sont créés aléatoirement à l'aide de la fonction *random*, définie par Matlab. Cette fonction permet de générer des nombres compris entre zéro et un, de manière aléatoire. Ensuite, le vecteur $b \in \mathbb{R}^m$ est engendré par combinaison linéaire des colonnes de A de telle sorte qu'il existe au moins une solution admissible au système $Ax = b$:

$$b = \sum_{i=1}^{n} \beta_i a_i, \tag{3.3}$$

où $a_i \in \mathbb{R}^m$ représente la $i^{\text{ème}}$ colonne de la matrice A. Les variables du type de problèmes que nous générons étant soumises à une contrainte de positivité, les coefficients β_i se doivent de la vérifier également, afin de constituer un point admissible du problème engendré. Notons que pour un certain nombre de problèmes, ces coefficients ont été introduits par l'utilisateur. Une fois que le nombre de variables est devenu grand, la fonction random à été utilisée afin de les sélectionner, pour des raisons de simplicité. Nous analyserons par la suite les observations extraites à partir des tests effectués sur chacun de ces problèmes.

3.3 Robustesse du code développé

Avant d'effectuer différents tests sur l'ensemble des problèmes créés, nous nous sommes assurés de la robustesse de notre code. Pour cela, une dizaine de problèmes, formulés en AMPL, un langage de modélisation pour la Programmation Mathématique (« A Modeling Language for Mathematical Programming ») [5], ont été soumis à des solveurs repris sur le serveur d'optimisation NEOS, à l'adresse internet donnée par [8]. Nous noterons parmis ceux-ci, le solveur MOSEK, ou encore KNITRO, qui implémentent une méthode primale-duale de points intérieurs en programmation linéraire, resp. non linéaire. Nous avons ainsi pu constater que la solution optimale primale-duale, obtenue pour chaque problème, correspondait à la solution fournie par notre implémentation de l'algorithme MPC.

3.4 Tests numériques

Partant tour à tour d'un point (x^0, λ^0, s^0) obtenu à partir d'une des trois méthodes de calcul du point de départ, nous avons donc appliqué l'algorithme MPC à un ensemble de problèmes, générés en suivant le schéma décrit à la

Section 3.2. Rappelons que les trois méthodes utilisées pour calculer le point de départ sont celle de Mehrotra, de Gondzio et ses confrères, et de Zhang.

Dans un premier tableau, Tab 3.2, chaque problème généré est repris, ainsi que le nombre de variables et de contraintes d'égalité qui lui sont associées. De cette manière, ces données ne seront plus rappelées dans les différents tableaux de résultats qui suivront. Notons que l'ensemble des problèmes à résoudre a été séparé en deux gammes. La première reprend les problèmes dont les coefficients de la combinaison linéaire (3.3) ont été déterminés par l'utilisateur, tandis que la deuxième contient ceux pour lesquels ces coefficients ont été générés aléatoirement. Par la suite, nous verrons quelle motivation a poussé à cette distinction.

Gamme 1			Gamme 2		
Problème	♯ var	♯ contr	Problème	♯ var	♯ contr
1	50	45	14	44	26
2	70	50	15	79	53
3	60	35	16	171	139
4	25	13	17	284	166
5	65	34	18	500	341
6	100	67	19	356	250
7	34	26	20	189	78
8	80	58	21	232	160
9	48	43	22	145	117
10	110	77	23	194	156
11	124	96	24	300	150
12	153	127	25	400	288
13	200	148			

TAB. 3.2 – Ensemble des problèmes tests.

Ayant défini l'ensemble des problèmes tests [TAB. 3.2], différentes caractéristiques des points de départ, engendrés pour chacun de ces problèmes à partir des trois approches présentées, sont analysées.

3.4.1 Analyse du point de départ de l'algorithme MPC

Tout d'abord, de par les contraintes de positivité imposées aux composantes x et s, le point de départ (x^0, λ^0, s^0) se doit de satisfaire ces conditions. Au cours des tests, nous avons pu observer que celles-ci étaient bien satisfaites pour les problèmes appartenant aux deux gammes, quelle que soit la

stratégie de départ utilisée. Ensuite, nous avons vu au début du deuxième chapitre qu'un bon point de départ devrait satisfaire deux conditions. La première est que ce point soit bien centré, i.e., que les produits $x_i^0 s_i^0$ soient similaires, pour $i = 1, ..., n$. Seulement, comme nous traitons des problèmes ayant jusqu'à 500 variables, analyser ces produits peut s'avérer assez long. C'est pourquoi nous avons choisi de calculer la mesure de proximité pour déterminer le caractère de centralité du point de départ, mesure définie par la relation (1.23) du premier chapitre et reprise ci-dessous pour des raisons de clarté :

$$\frac{1}{\mu}\|XSe - \mu e\| = \frac{1}{\mu}\left\| \begin{bmatrix} x_1 s_1 \\ \vdots \\ x_n s_n \end{bmatrix} - \left(\frac{x^T s}{n}\right) e \right\|. \tag{3.4}$$

De par sa définition, plus cette mesure est petite, plus le point de départ est bien centré par rapport à $\mathcal{C}$. La seconde condition porte quant à elle sur le caractère de non-admissibilité relatif au point de départ, quantifié à l'aide de la relation (2.1), à savoir :

$$\frac{\|(r_b^0, r_c^0)\|}{\mu_0}. \tag{3.5}$$

Selon S.J. Wright [12, page 224], pour que le point soit considéré comme un bon point de départ, cette valeur ne doit pas être trop grande. Par la suite, nous dirons qu'un point est « plus réalisable » qu'un autre, si la valeur de l'expression (3.5) qui lui est relative, est plus petite. Notons que pour les tests, la norme-2 est utilisée lors de l'évaluation de la mesure de proximité, comme du caractère de non-admissibilté. Ainsi, ces deux caractéristiques sont évaluées pour les problèmes de chaque gamme, les résultats obtenus étant repris et commentés au cours des pages suivantes.

Dans un premier temps, les problèmes de la Gamme 1 sont traités, en commençant par analyser le caractère de non-admissibilité associé aux points de départ calculés par les trois approches, pour chaque problème. La table TAB. 3.3 présente les résultats obtenus.

Gamme 1			
Problèmes	$\frac{\|(r_b^0, r_c^0)\|_2}{\mu_0}$		
	Mehrotra	Gondzio	Zhang
1	$2.77\ 10^2$	$1.09\ 10^1$	$2.37\ 10^1$
2	$1.14\ 10^2$	5.31	$6.83\ 10^1$
3	$1.07\ 10^2$	5.55	$4.13\ 10^1$
4	$0.33\ 10^2$	0.53	9.92
5	$8.59\ 10^1$	6.28	$6.80\ 10^1$
6	$2.67\ 10^2$	$3.07\ 10^1$	$1.72\ 10^2$
7	$6.10\ 10^1$	$1.17\ 10^1$	$5.11\ 10^1$
8	$2.19\ 10^2$	4.17	$7.79\ 10^1$
9	$1.39\ 10^2$	3.58	$2.42\ 10^1$
10	$1.86\ 10^2$	4.79	$1.16\ 10^2$
11	$2.66\ 10^2$	9.80	$1.51\ 10^2$
12	$9.12\ 10^2$	$2.55\ 10^1$	$3.47\ 10^2$
13	$5.67\ 10^2$	$4.83\ 10^1$	$3.37\ 10^2$

TAB. 3.3 – Caractère de non-admissibilité du point de départ de l'algorithme MPC pour la résolution des problèmes de la Gamme 1.

L'analyse de la valeur du caractère de non-admissibilité nous montre que celle-ci est grande pour un point obtenu par la méthode de Mehrotra, alors qu'elle est plus petite dans les deux autres cas. En effet, elle est en général de l'ordre de la centaine pour la méthode de Mehrotra, de l'ordre de la dizaine ou de la centaine pour Zhang, tandis que pour Gondzio, elle s'évalue autour de la dizaine. Nous en déduisons donc que l'approche de Gondzio produit un point de départ plus réalisable pour le problème à résoudre, que celui produit par l'approche de Zhang et encore plus que le point obtenu par la méthode de Mehrotra.

Ensuite, afin de comparer le caractère de centralité des points de départ, la table TAB. 3.4 reprend les valeurs de la mesure de proximité, obtenues pour chaque point calculé.

Gamme 1			
Problèmes	$\frac{\|X^0S^0e-\mu_0e\|_2}{\mu_0}$		
	Mehrotra	Gondzio	Zhang
1	4.46	8.76	$1.05\ 10^1$
2	2.62	4.10	$1.08\ 10^1$
3	2.91	5.84	$1.17\ 10^1$
4	2.05	2.72	6.40
5	2.36	5.02	$1.30\ 10^1$
6	3.93	$1.01\ 10^1$	$1.50\ 10^1$
7	2.95	7.72	7.20
8	2.47	4.17	9.85
9	3.65	5.93	6.99
10	3.35	4.84	$1.17\ 10^1$
11	4.31	7.51	$1.35\ 10^1$
12	5.01	$1.27\ 10^1$	$1.62\ 10^1$
13	4.89	$1.23\ 10^1$	$2.10\ 10^1$

TAB. 3.4 – Mesure de proximité du point de départ de l'algorithme MPC pour la résolution des problèmes de la Gamme 1.

D'une part, il suit de ces résultats que l'approche de Mehrotra calcule toujours un point de départ de l'algorithme MPC plus proche de $\mathcal{C}$ que les deux autres méthodes. D'autre part, nous remarquons également que le point obtenu en appliquant la stratégie de Zhang est moins bien centré que celui déterminé par la méthode de Gondzio, pour le problème correspondant. En effet, la mesure de proximité associée à ce point est plus grande que celle associée au point obtenu par la méthode de Gondzio.

Dans un deuxième temps, nous avons effectué des tests similaires sur les problèmes de la Gamme 2, l'analogue des tables TAB. 3.3 et TAB. 3.4 étant donné, respectivement, par les tables TAB. 3.5 et TAB. 3.6.

Gamme 2			
Problèmes	$\frac{\|(r_b^0, r_c^0)\|_2}{\mu_0}$		
	Mehrotra	Gondzio	Zhang
14	$3.56\ 10^1$	$5.71\ 10^1$	$6.31\ 10^1$
15	$9.76\ 10^1$	$1.46\ 10^2$	$1.69\ 10^2$
16	$6.77\ 10^2$	$4.90\ 10^2$	$5.49\ 10^2$
17	$5.27\ 10^2$	$9.18\ 10^2$	$1.22\ 10^3$
18	$2.70\ 10^3$	$2.36\ 10^3$	$3.02\ 10^3$
19	$1.66\ 10^3$	$1.43\ 10^3$	$1.83\ 10^3$
20	$1.92\ 10^2$	$4.38\ 10^2$	$5.40\ 10^2$
21	$4.93\ 10^2$	$7.14\ 10^2$	$9.08\ 10^2$
22	$5.63\ 10^2$	$3.95\ 10^2$	$4.57\ 10^2$
23	$5.87\ 10^2$	$6.07\ 10^2$	$6.70\ 10^2$
24	$7.24\ 10^2$	$9.03\ 10^2$	$1.23\ 10^3$
25	$1.65\ 10^3$	$1.75\ 10^3$	$2.15\ 10^3$

TAB. 3.5 – Caractère de non-admissibilité du point de départ de l'algorithme MPC pour la résolution des problèmes de la Gamme 2.

Le caractère de non-admissibilité des points de départ générés pour la Gamme 2 [TAB. 3.5] est en général élevé pour chacune des approches utilisées. Néanmoins, l'approche de Mehrotra calcule, en moyenne, un point plus réalisable que les deux autres approches. Notons que pour un même problème, le point de départ obtenu par la méthode de Zhang satisfait moins bien, en moyenne, les contraintes de ce problème, qu'un point déterminé par l'approche de Gondzio. Autrement dit, un point calculé par la stratégie de Gondzio est, en moyenne, plus réalisable qu'un point donné par l'approche de Zhang, mais moins réalisable qu'un point déterminé par la méthode de Mehrotra. De plus, en comparant les tables TAB. 3.3 et TAB. 3.5, nous remarquons que, pour un même problème, la différence entre les valeurs attribuées au caractère de non-admissibilité des points de départ obtenus pour la Gamme 2 est moins frappante que pour la Gamme 1.

Gamme 2			
Problèmes	$\frac{\|X^0S^0e-\mu_0e\|_2}{\mu_0}$		
	Mehrotra	Gondzio	Zhang
14	2.12	0.16	1.48
15	3.07	0.38	3.34
16	4.66	0.38	4.81
17	5.43	0.53	$1.27\ 10^1$
18	6.47	0.25	$1.53\ 10^1$
19	4.75	0.67	$1.35\ 10^1$
20	4.60	0.48	9.60
21	4.94	0.56	7.85
22	3.81	0.33	4.25
23	4.75	0.28	5.03
24	5.01	0.50	$1.68\ 10^1$
25	6.02	0.67	$1.17\ 10^1$

TAB. 3.6 – Mesure de proximité du point de départ de l'algorithme MPC pour la résolution des problèmes de la Gamme 2.

Par contre, nous observons que le point de départ déterminé à partir de la méthode de Gondzio se trouve toujours plus proche de $\mathcal{C}$ que les points engendrés par les deux autres approches [TAB. 3.6]. Cependant, soulignons que l'approche de Mehrotra engendre un point mieux centré que celui fourni par l'approche de Zhang, excepté pour le problème 14.

3.4.2 Résultats généraux

Chacun des problèmes tests est alors résolu entièrement par l'algorithme MPC, afin de comparer le nombre d'itérations nécessaire pour trouver une solution, selon l'approche de calcul du point de départ sélectionnée. Les tableaux TAB. 3.7 et TAB. 3.8 contiennent les résultats obtenus.

Gamme 1			
Problème	♯ d'itérations		
	Mehrotra	Gondzio	Zhang
1	6	6	6
2	7	7	8
3	7	7	9
4	7	7	9
5	7	7	11
6	9	8	10
7	6	6	9
8	7	7	8
9	6	5	7
10	8	8	10
11	10	10	11
12	10	9	11
13	10	11	14

TAB. 3.7 – Nombre d'itérations nécessaire pour résoudre les problèmes de la Gamme 1.

A partir des résultats repris dans la table TAB. 3.7 pour la première gamme de problèmes, nous pouvons observer que lorsque le point de départ de l'algorithme MPC est calculé par l'approche de Zhang, la résolution du problème nécessite, en général, plus d'itérations. En effet, la résolution du premier problème est effectuée en un même nombre d'itérations, quelle que soit la méthode de calcul du point de départ utilisée. Par contre, partant d'un point donné par la méthode de Mehrotra ou de Gondzio, ce nombre est identique, à une itération près pour certains problèmes, tandis qu'il est toujours plus élevé en utilisant l'approche de Zhang. Soulignons que dans le cas où le nombre d'itérations n'est pas le même pour Mehrotra et Gondzio, l'application de l'algorithme MPC à partir d'un point déterminé par la méthode de Gondzio, permet de résoudre le problème donné en utilisant une itération de moins, sauf pour le problème 13.

Gamme 2			
Problème	♯ d'itérations		
	Mehrotra	Gondzio	Zhang
14	7	7	7
15	7	8	8
16	11	11	10
17	9	10	11
18	11	11	13
19	11	11	12
20	9	9	10
21	9	9	10
22	8	8	9
23	8	9	9
24	9	9	9
25	10	11	12

TAB. 3.8 – Nombre d'itérations nécessaire pour résoudre les problèmes de la Gamme 2.

Cependant, nous observons des résultats quelque peu différents en ce qui concerne la deuxième gamme de problèmes [TAB. 3.8]. En effet, partant d'un point calculé à l'aide de l'approche de Mehrotra ou de Gondzio, l'algorithme MPC résoud les problèmes de la deuxième gamme en un nombre d'itérations identique, à l'exception de quatre problèmes. Dans ces quatre cas, la résolution des problèmes concernés est effectuée, par l'algorithme MPC, en une itération de moins si la méthode de Mehrotra est appliquée au départ de celui-ci. De plus, nous observons de nouveau que, si le point de départ est déterminé par l'approche de Zhang, la résolution des problèmes de cette gamme nécessite plus d'itérations, à quelques exceptions près. Toutefois, cette différence entre le nombre d'itérations nécessaire à l'algorithme MPC est moins frappante que pour les problèmes de la Gamme 1.

Par conséquent, l'application de l'algorithme MPC pour résoudre l'ensemble des problèmes de la Gamme 1 et 2, semble moins efficace lorsque le point de départ est calculé par la méthode de Zhang, puisque, dans ce cas, le nombre d'itérations nécessaire est en général plus grand.

Les différents tests ayant été effectués, nous pouvons tirer quelques conclusions quant à l'efficacité des trois approches de calcul du point de départ, en reprenant également les résultats commentés au cours de la Section 3.4.1.

3.4.3 Conclusions

A partir des différentes tables de résultats présentées et analysées précédemment, nous avons remarqué qu'en pratique, peu de points satisfont simultanément les deux caractéristiques suggérées par S.J. Wright, [12, page 224], pour être considéré comme un « bon » point, i.e., un point tel qu'au départ de celui-ci, l'algorithme MPC résoud le problème associé en nécessitant moins d'itérations. Rappelons que ces deux propriétés sont d'être un point bien centré et dont la valeur représentant le caractère de non-admissibilité soit petite.

Pour les problèmes appartenant à la Gamme 1, les points de départ sont plus réalisables lorsqu'ils sont obtenus à partir de l'approche de Gondzio, tandis qu'ils sont mieux centrés s'ils sont calculés par la méthode de Mehrotra. Cependant, l'approche de Mehrotra engendre des points moins réalisables que les deux autres méthodes. En ce qui concerne l'approche de Zhang, les points obtenus à l'aide de celle-ci ne s'avèrent pas bien centrés, en comparaison avec les points déterminés par les deux autres méthodes. Ces points sont en plus moins réalisables que ceux générés par la méthode de Gondzio, mais satisfont mieux les contraintes d'égalité des problèmes concernés que les points calculés par la méthode de Mehrotra. Soulignons également que l'algorithme MPC résoud les problèmes de cette gamme en nécessitant sensiblement plus d'itérations si le point de départ est obtenu à partir de l'approche de Zhang, tandis qu'il en nécessite moins au départ d'un point calculé par les méthodes de Gondzio ou de Mehrotra. De plus, nous avons constaté au cours de l'analyse de la TAB. 3.7, que dans le cas où le point de départ est déterminé par l'approche de Gondzio ou de Mehrotra, le nombre d'itérations nécessaire à l'algorithme est identique, à une itération près. Par conséquent, il semble que les points obtenus en appliquant la méthode de Gondzio soient « meilleurs » que les points de départ calculés par l'approche de Mehrotra. De plus, les points générés par la méthode de Zhang pourraient être qualifiés de « moins bons » points de départ.

Pour la Gamme 2, les points de départ calculés par l'approche de Zhang sont des points qui satisfont moins bien les *deux* propriétés. En effet, ces points sont toujours moins réalisables et moins bien centrés par rapport à $\mathcal{C}$, en comparaison aux points générés par les deux autres méthodes. De plus, nous avons observé que l'algorithme MPC nécessite plus d'itérations en partant de ces points. Par contre, l'approche de Mehrotra détermine des points de départ plus réalisables, en moyenne, que la méthode de Gondzio, tandis qu'ils sont moins bien centrés. Partant de ces points, l'algorithme MPC résoud les problèmes associés en prenant autant d'itérations que s'il

débute avec les points engendrés par Gondzio, à l'exception de quelques problèmes en faveur de Mehrotra, comme nous le montre la TAB. 3.8. Par conséquent, il semble que les points générés par la stratégie de Zhang soient de « moins bons » points de départ, tandis que ceux engendrés par l'approche de Mehrotra ou de Gondzio semblent « meilleurs ».

La table TAB. 3.9, présentée ci-dessous, résume les observations obtenues à partir des différents tests effectués, en classant les méthodes à l'aide des numéros 1, 2 ou 3, séparément pour les deux gammes de problèmes. Ainsi, le numéro 1 sera attribuée à la méthode qui détermine un point de départ satisfaisant le mieux le critère de non-admissibilité ou le critère de centralité, placés verticalement. Ce numéro sera également affecté à une méthode telle que le nombre d'itérations nécessaire à l'algorithme MPC est meilleur, au départ d'un point déterminé par celle-ci.

	Mehrotra	Gondzio	Zhang
Gamme 1			
$\frac{\|(r_b^0, r_c^0)\|_2}{\mu^0}$ est petit	3	1	2
$\frac{\|X^0 S^0 e - \mu_0 e\|_2}{\mu_0}$ est petite	1	2	3
♯ itérations est meilleur	2	1	3
Gamme 2			
$\frac{\|(r_b^0, r_c^0)\|_2}{\mu^0}$ est petit	1	2	3
$\frac{\|X^0 S^0 e - \mu_0 e\|_2}{\mu_0}$ est petite	2	1	3
♯ itérations est meilleur	1	2	3

TAB. 3.9 – Classement des méthodes de calcul du point de départ de l'algorithme MPC.

Au vu du classement présenté par la table TAB. 3.9, il semble que nous puissions en déduire que le caractère de non-admissibilité d'un point de départ soit un facteur important dans la détermination du fait que ce point soit un « bon » point de départ. En effet, alors que la méthode de Mehrotra passe de la troisième position à la première si l'on considère le critère de non-admissibilité, respectivement pour la Gamme 1 et la Gamme 2, nous remarquons d'une part que du point de vue du nombre d'itérations nécessaire à l'algorithme MPC au départ d'un point déterminé par cette méthode, celle-ci se positionne en première place pour la Gamme 2, tandis qu'elle se trouve en deuxième position pour la Gamme 1, et cela malgré le fait qu'elle régresse d'une position lorsque le critère de centralité est pris en compte pour la Gamme 2. D'autre part, nous constatons que pour la Gamme 2, la méthode de Zhang se trouve toujours en dernière position, quel que soit le critère analysé. Cependant, nous avions observé que l'écart entre les valeurs représentant le caractère de non-admissibilité, obtenues pour les trois méthodes, est moins grand pour la Gamme 2 que pour la Gamme 1, de même que la différence entre le nombre d'itérations nécessaire à l'algorithme MPC. Par conséquent, il semble que le caractère de non-admissibilité d'un point de départ soit un facteur dominant dans la caractérisation d'un « bon » point de départ, bien que le caractère de centralité ne soit pas à négliger.

De cette manière, nous clôturons la première partie de ce mémoire, partie dont l'objectif consistait à décrire et analyser différentes approches de calcul de points de départ pour des algorithmes de points intérieurs en programmation linéaire. La deuxième partie est, quant à elle, consacrée dans un premier temps à la description d'une méthode barrière de points intérieurs en programmation non linéaire. Dans un deuxième temps, une nouvelle approche de calcul de points de départ sera présentée dans ce cadre, puis adaptée au cadre linéaire en vue d'effectuer une comparaison avec les trois approches exposées au deuxième chapitre et testées au cours de celui-ci.

Deuxième partie

Programmation Non Linéaire

Chapitre 4

Une méthode de Points Intérieurs en Programmation Non Linéaire pour des Problèmes à Grande Echelle

4.1 Introduction

Ce chapitre a pour objectif de décrire une méthode barrière de points intérieurs, introduite par R.H. Byrd, M.E. Hribar et J. Nocedal [3] et permettant de résoudre des problèmes d'optimisation non linéaires de grande taille du type :

$$\begin{array}{ll} min & f(x) \\ s.c.: & h(x) = 0 \\ & g(x) \leq 0, \end{array} \tag{4.1}$$

où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $h : \mathbb{R}^n \longrightarrow \mathbb{R}^t$ et $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ sont des fonctions régulières et où n est grand. Cette méthode de points intérieurs peut être qualifiée d'approche primale-duale, mais également d'approche primale. Au cours du chapitre, nous justifierons dans quelle situation l'une ou l'autre approche est définie. D'autre part, notons que, comme ce fut le cas dans le cadre de la programmation linéaire, nous ne nous préoccuperons pas, dans ce chapitre, du calcul du point de départ de la méthode.

4.2 Méthode barrière

L'idée de base de la méthode barrière décrite dans [3] est la suivante. A chaque itération, un **sous-problème barrière** sous contraintes d'égalité est résolu approximativement à l'aide d'une itération de Programmation Quadratique Séquentielle (PQS) utilisant les régions de confiance. Ce dernier est de la forme suivante :

$$\begin{array}{rc} min_{x,s} & f(x) - \mu \sum_{i=1}^{m} lns_i \\ s.c. : & h(x) = 0 \\ & g(x) + s = 0, \end{array} \tag{4.2}$$

où $\mu > 0$ est appelé le **paramètre barrière** et où la variable d'écart s est supposée être strictement positive dans $\mathbb{R}^m$. Afin de caractériser la solution du sous-problème barrière (4.2), le Lagrangien associé à celui-ci est introduit :

$$\mathcal{L}(x, s, \lambda_h, \lambda_g) = f(x) - \mu \sum_{i=1}^{m} lns_i + \lambda_h^T h(x) + \lambda_g^T (g(x) + s), \tag{4.3}$$

où λ_h et λ_g sont les multiplicateurs de Lagrange, associés respectivement aux contraintes de (4.2). La solution, obtenue à l'issue d'une itération PQS et notée $(\hat{x}, \hat{s})$, est un point approché de la solution de (4.2) dans le sens où la mesure des conditions d'optimalité, désignée par E, est inférieure à un seuil fixé ϵ_μ, i.e., $E(\hat{x}, \hat{s}; \mu) \leq \epsilon_\mu$, où

$$\begin{aligned} E(x, s; \mu) = max(&||\nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g||_\infty, ||S\lambda_g - \mu e||_\infty, \\ &||h(x)||_\infty, ||g(x) + s||_\infty) \end{aligned} \tag{4.4}$$

et où ϵ_μ est la tolérance déterminant l'exactitude de la solution des sous-problèmes barrières. Soulignons que les notations utilisées dans la définition de E en (4.4) et définies ci-après, seront valables pour la suite :

$$\begin{aligned} e &= (1, ..., 1)^T \in \mathbb{R}^m, \\ S &= diag(s_1, ..., s_m) \in \mathbb{R}^{m \times m}, \\ A_h(x) &= [\nabla h_1(x), ..., \nabla h_t(x)] \in \mathbb{R}^{n \times t}, \\ A_g(x) &= [\nabla g_1(x), ..., \nabla g_m(x)] \in \mathbb{R}^{n \times m}, \end{aligned}$$

où A_h est supposé de rang t. Nous verrons par la suite que les termes définissant la mesure d'optimalité correspondent en fait à chacune des équations du système KKT perturbé sur lequel l'algorithme de points intérieurs est basé.

Par ailleurs, les itérés (x^k, s^k), engendrés au cours de la méthode barrière, sont des points strictement positifs en la variable s, qui ne vérifient pas nécessairement les contraintes du problème (4.1). En programmation linéaire, nous avions déjà rencontré de tels algorithmes, engendrant des itérés strictement positifs en certaines variables, mais pas nécessairement réalisables [Chapitre 1].

Revenant au problème (4.2), l'objectif de la méthode barrière est de faire converger le paramètre μ vers zéro de sorte que la suite des solutions des sous-problèmes (4.2) tende vers un point stationnaire du problème de départ (4.1). Pour cela, μ est réduit à chaque itération d'un facteur constant θ appartenant à l'intervalle ouvert $(0, 1)$. De la même manière, la tolérance ϵ_μ est diminuée de ce facteur lors du passage d'une itération à l'autre, convergeant ainsi vers zéro. La description de cette méthode barrière de points intérieurs se termine en résumant les différentes étapes suivies par celle-ci à l'aide de l'algorithme suivant, le schéma de cet algorithme étant suivi afin de résoudre un problème non linéaire du type (4.1).

Algorithme barrière

Choisir une valeur initiale pour $\mu > 0$;

Sélectionner les paramètres $\epsilon_\mu > 0$, $\theta \in (0, 1)$ et la tolérance d'arrêt final ϵ_{TOL} ;

Choisir le point de départ x^0 et $s^0 > 0$;

Evaluer la fonction objectif, les contraintes et leurs dérivées au point x^0.

Répéter jusqu'à ce que $E(x, s; 0) \leq \epsilon_{\text{TOL}}$:

1. appliquer une méthode PQS avec régions de confiance, à partir de (x^k, s^k), pour trouver une solution approchée $(\hat{x}, \hat{s})$ du problème barrière (4.2) satisfaisant $E(\hat{x}, \hat{s}; \mu) \leq \epsilon_\mu$;

2. poser $\mu \longleftarrow \theta\mu$, $\epsilon_\mu \longleftarrow \theta\epsilon_\mu$, $x^k \longleftarrow \hat{x}$, $s^k \longleftarrow \hat{s}$, $k \longleftarrow k + 1$;

Fin (boucle).

L'étape la plus importante de cet algorithme est effectuée lors du premier pas. Les sections suivantes sont consacrées à la description complète d'une itération PQS.

4.3 Une itération de Programmation Quadratique Séquentielle

Comme nous l'avons introduit au début de ce chapitre, les sous-problèmes barrières (4.2) sont résolus à l'aide d'une méthode PQS. Afin de suivre un raisonnement clair, la description de celle-ci sera directement appliquée à un problème du type (4.2), à partir de l'itéré (x^k, s^k).

Pour commencer, chaque itération d'une méthode PQS construit un modèle quadratique du Lagrangien (4.3), moyennant la connaissance des multiplicateurs de Lagrange λ_h et λ_g. Le détail des calculs à partir desquels ces multiplicateurs sont obtenus, est repris à la fin de cette section. Ensuite, un pas de l'algorithme est calculé en minimisant ce modèle. Ce pas est contraint de satisfaire une approximation linéaire des contraintes du problème (4.2) et est soumis à une région de confiance notée T_k. De cette manière, le sous-problème résolu par une itération PQS prend la forme suivante :

$$\begin{aligned} min_{d_x,d_s} \quad & \nabla f(x^k)^T d_x + \tfrac{1}{2} d_x^T \nabla^2_{xx} \mathcal{L}(x^k, s^k, \lambda_h, \lambda_g) d_x - \mu e^T S_k^{-1} d_s \quad (4.5a) \\ & + \tfrac{1}{2} d_s^T \Sigma_k d_s \\ s.c. : \quad & A_h(x^k)^T d_x + h(x^k) = r_h \quad (4.5b) \\ & A_g(x^k)^T d_x + d_s + g(x^k) + s^k = r_g \quad (4.5c) \\ & (d_x, d_s) \in T_k, \quad (4.5d) \end{aligned}$$

où $\Sigma_k \in \mathbb{R}^{m \times m}$ est une matrice diagonale définie positive représentant le Hessien du Langrangien (4.3) par rapport à s ou une approximation de celui-ci et où les vecteurs r_h et r_g définissent les vecteurs résidus qui représentent respectivement la valeur par laquelle les contraintes suivantes :

$$\begin{aligned} A_h(x^k)^T d_x + h(x^k) &= 0, \quad (4.6a) \\ A_g(x^k)^T d_x + d_s + g(x^k) + s^k &= 0, \quad (4.6b) \end{aligned}$$

sont violées. Le choix de $\Sigma_k \in \mathbb{R}^{m \times m}$ est important car il détermine si l'itération est une itération primale ou primale-duale. Avant de s'arrêter quelques instants sur le fait de savoir dans quel cas nous sommes, définissons la **région de confiance** T_k, et cela dans le but d'accomplir deux objectifs. D'une part, cette région définit un ensemble « de confiance » autours du point x^k, dans lequel le modèle quadratique et les contraintes linéarisées peuvent être considérés comme de bonnes approximations du problème (4.2). Les pas obtenus à l'issue de la résolution du problème (4.5a)-(4.5d) sont ainsi limités à l'intérieur de celui-ci :

$$||(d_x, S_k^{-1} d_s)||_2 \leq \Delta_k, \quad (4.7)$$

où Δ_k est le rayon de T_k et est choisi pour être plus grand que un. La mise à échelle des pas d_s, introduite dans la relation (4.7), a pour but de les pénaliser aux abords de la frontière de T_k, puisque les variables d'écart s_i^k, $i = 1, ..., m$, ne peuvent approcher zéro prématurément. D'autre part, la région de confiance assure que ces variables d'écart vérifient strictement la contrainte de non-négativité, en imposant la condition suivante :

$$d_s \geq -\tau s^k, \tag{4.8}$$

où $\tau \in (0, 1)$. Finalement, la région de confiance ainsi définie garantit au problème (4.5a)-(4.5d) une solution finie, même si le Hessien $\nabla^2_{xx}\mathcal{L}(x^k, s^k, \lambda_h, \lambda_g)$ n'est pas défini positif [3, page 881].

Revenons à la définition de la matrice Σ_k. Comme il a été cité précédemment, le choix de celle-ci est déterminant pour caractériser la méthode. Le premier choix qui peut être effectué est de considérer le Hessien de $\mathcal{L}$ par rapport à s, i.e., $\Sigma_k = \nabla^2_{ss}\mathcal{L}_k$:

$$\Sigma_k = \mu S_k^{-2}. \tag{4.9}$$

Afin d'étudier l'impact de Σ_k dans le calcul du pas, analysons le cas où les hypothèses suivantes sont satisfaites :

- la matrice $\nabla^2_{xx}\mathcal{L}_k$ est définie positive sur le noyau du gradient des contraintes,
- le vecteur résidu $r = (r_h, r_g)$ est nul,
- le pas généré par le problème (4.5a)-(4.5d) se trouve dans l'intérieur strict de T_k.

Sous ces hypothèses, le sous-problème quadratique (4.5a)-(4.5d) admet une solution unique, notée $d = (d_x, d_s)$, satisfaisant le système :

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}_k & 0 & A_h(x^k) & A_g(x^k) \\ 0 & \Sigma_k & 0 & I \\ A_h^T(x^k) & 0 & 0 & 0 \\ A_g^T(x^k) & I & 0 & 0 \end{pmatrix} \begin{pmatrix} d_x \\ d_s \\ \lambda_h^+ \\ \lambda_g^+ \end{pmatrix} = \begin{pmatrix} -\nabla f(x^k) \\ \mu S_k^{-1} e \\ -h(x^k) \\ -g(x^k) - s^k \end{pmatrix}. \tag{4.10}$$

Si Σ_k est définie par (4.9), cette approche est appelée **approche primale**. Dans ce cas, il est assez évident de voir que la résolution du système (4.10) équivaut à appliquer une itération de Newton aux conditions KKT associées au sous-problème (4.2) et données par :

$$\nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g = 0, \tag{4.11a}$$

$$-\mu S^{-1} e + \lambda_g = 0, \tag{4.11b}$$

$$h(x) = 0, \tag{4.11c}$$

$$g(x) + s = 0. \tag{4.11d}$$

Toutefois, cette approche ne garantit pas la positivité stricte des variables d'écart, celles-ci pouvant devenir négatives et entraîner une méthode inefficace [3, page 882]. Dès lors, pour soulever cette difficulté, le système (4.11a)-(4.11d) est modifié en un système KKT dit perturbé, dont le membre de gauche a été évoqué lors de la définition (4.4) de la mesure des conditions d'optimalité E :

$$\begin{aligned} \nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g &= 0, && (4.12a)\\ S\lambda_g - \mu e &= 0, && (4.12b)\\ h(x) &= 0, && (4.12c)\\ g(x) + s &= 0, && (4.12d) \end{aligned}$$

système obtenu en multipliant l'équation (4.11b) par la matrice S. Malgré que le sytème (4.12a)-(4.12d) ait la même solution que (4.11a)-(4.11d), l'application de la méthode de Newton à (4.12a)-(4.12d) produit des itérés différents. En effet, un pas de Newton appliqué à celui-ci est donné par la solution du système (4.10) où

$$\Sigma_k = S_k^{-1}\Lambda_g. \tag{4.13}$$

Soulignons que la matrice Λ_g est une matrice diagonale contenant les multiplicateurs de Lagrange associés aux contraintes d'inégalité $g_i(x) \leq 0$, pour $i = 1, ..., m$. Par conséquent, l'approche résolvant le système (4.10) où Σ_k est définie par (4.13), consiste en une **approche primale-duale**. Ce deuxième choix de Σ_k peut être vu comme une approximation du Hessien $\nabla^2_{ss}\mathcal{L}$. En effet, substituant dans la définition (4.9) l'équation (4.11b) satisfaite à la solution (x, s, λ) du problème (4.2), à savoir $\mu S^{-1} = \Lambda_g$, la relation (4.13) est obtenue :

$$\Sigma_k = S_k^{-1}(\mu S_k^{-1}) = S_k^{-1}\Lambda_g.$$

L'avantage du système (4.12a) -(4.12d) est que les dérivées de sa deuxième équation sont bornées lorsque les variables d'écart s_i^k convergent vers zéro, pour $i = 1, ..., m$. De plus, la méthode primale-duale empêche les variables d'écart de devenir négatives et permet en général d'effectuer de bons progrès vers la solution [3, page 883].

Une fois la matrice Σ_k choisie par la méthode PQS, le pas $d = (d_x, d_s)$ est calculé par résolution approximative du problème (4.5a)-(4.5d) en suivant une approche que nous définirons au cours de la prochaine section. Finalement, la **fonction de mérite**

$$\phi(x, s; \nu) = f(x) - \mu \sum_{i=1}^{m} lns_i + \nu \left\| \begin{bmatrix} h(x) \\ g(x) + s \end{bmatrix} \right\|_2, \tag{4.14}$$

où $\nu > 0$ est un paramètre de pénalité, entre en compte afin de déterminer si le pas obtenu est acceptable ou non. En effet, si d produit une réduction suffisante de ϕ, celui-ci est utilisé pour obtenir le nouvel itéré, sinon il est rejeté. L'algorithme suivant vient ainsi clôturer la description d'une itération PQS, permettant de résoudre les problèmes du type (4.2).

Algorithme PQS de région de confiance

Considérer les paramètres donnés $\mu > 0$, $\epsilon_\mu > 0$ et les valeurs de k, x^k et $s^k > 0$;

Définir la région de confiance T_k ;

Calculer les multiplicateurs de Lagrange λ_h et λ_g ;

Répéter jusqu'à ce que $E(x^k, s^k; \mu) \leq \epsilon_\mu$:

1. calculer $d = (d_x, d_s)$ par résolution approximative du problème (4.5a)-(4.5d) ;

2. *si* le pas d fournit une diminution suffisante dans ϕ, *alors* :
   poser $x^{k+1} = x^k + d_x$ et $s^{k+1} = s^k + d_s$;
   calculer les nouveaux multiplicateurs de Lagrange λ_h et λ_g ;
   élargir T_k dans la mesure du possible ;

   sinon :
   poser $x^{k+1} = x^k$ et $s^{k+1} = s^k$;
   rétrécir T_k ;

   fin (si) ;

3. poser $k \longleftarrow k + 1$;

Fin (boucle).
$\hat{x} = x^k$ et $\hat{s} = s^k$.

Cet algorithme est appelé à chaque exécution du pas 1 de l'algorithme barrière [Section 4.2]. Avant de décrire de manière plus approfondie le calcul du pas $d = (d_x, d_s)$, solution du sous-problème quadratique (4.5a)-(4.5d),

terminons cette section en expliquant la manière dont les multiplicateurs de Lagrange sont calculés.

Utilisant une approche des moindres carrés, en anglais « least squares » (LS), le vecteur $\lambda = (\lambda_h, \lambda_g)$ est obtenu en minimisant la norme Euclidienne de (4.12a)-(4.12b). Ainsi, à l'itération k, λ est donné par :

$$\lambda_k = \begin{bmatrix} \lambda_h \\ \lambda_g \end{bmatrix} = \lambda^{\text{LS}}(x^k, s^k, \mu) = (\hat{A}_k^T \hat{A}_k)^{-1} \hat{A}_k^T \begin{bmatrix} -\nabla f(x^k) \\ \mu e \end{bmatrix}, \tag{4.15}$$

où

$$\hat{A}_k = \begin{bmatrix} A_h(x^k) & A_g(x^k) \\ 0 & S_k \end{bmatrix}. \tag{4.16}$$

Comme nous l'expliquerons dans la Section 4.6, le calcul de (4.15) s'effectue en résolvant un système augmenté. A l'issue du calcul de λ_k, il se peut que les multiplicateurs $[\lambda_g]_i$, $i = 1, ..., m$, soient à valeur négative. Soulignons que par la suite, nous utiliserons la notation $[..]_i$ pour désigner la $i^{\text{ème}}$ composante d'un vecteur déjà indicé. Dès lors, leur utilisation dans le choix primal-dual de Σ_k, (4.13), est remis en question. En effet, comme le Hessien du terme barrière $-\mu \sum_{i=1}^{m} lns_i$ est défini positif, il semble indésirable de créer une approximation indéfinie Σ_k de celui-ci. Pour remédier à cette difficulté, R.H. Byrd, M.E. Hribard et J. Nocedal ont suivi l'approche suivante [3]. Plutôt que d'assurer la positivité des $[\lambda_g]_i$, $i = 1, ..., m$, ils ont imposé au modèle quadratique de rester convexe dans les variables d'écart. De cette manière, le $i^{\text{ème}}$ élément diagonal de Σ_k, dans la version primale-duale de l'algorithme, est défini par :

$$[\sigma_k]_i = \begin{cases} [\lambda_g]_i / s_i & \text{si } [\lambda_g]_i > 0, \\ \mu/(s_i)^2 & \text{sinon.} \end{cases} \tag{4.17}$$

Ainsi, lorsqu'un multiplicateur $[\lambda_g]_i$, donné par (4.15), est négatif, l'entrée correspondante dans la matrice primale-duale Σ_k coïncide avec l'entrée correspondante dans le Hessien primal. Néanmoins, afin d'éviter des changements trop brusques dans Σ_k lorsque μ décroît, la définition de λ_k est légèrement modifiée dans le cas primal-dual. Dans ce cas, considérant le point de départ (x^k, s^k) d'un nouveau sous-problème barrière, i.e., se trouvant au départ de l'algorithme PQS, λ_g est pris, dans la relation (4.17), comme le multiplicateur obtenu à partir du dernier itéré du sous-problème barrière précédent.

Par conséquent, les multiplicateurs de Lagrange sont définis de la manière suivante :

$$\lambda_k = \begin{cases} \lambda^{\text{LS}}(x^k, s^k, \mu) & \text{dans la version primale,} \\ \lambda^{\text{LS}}(x^k, s^k, \overline{\mu}) & \text{dans la version primale-duale,} \end{cases} \tag{4.18}$$

où $\overline{\mu}$ est la valeur du paramètre barrière utilisé dans le calcul de (x^k, s^k).

4.4 Résolution du sous-problème quadratique

La résolution du sous-problème (4.5a)-(4.5d), défini lors d'une itération PQS, détermine le pas d de l'algorithme barrière [Section 4.2] qui est obtenu par combinaison d'un **pas normal** v et d'un **pas tangentiel** w.

4.4.1 Calcul du pas normal

Dans un premier temps, le pas normal est calculé afin de trouver une valeur du vecteur résidu $r = (r_h, r_g)$ rendant le sous-problème quadratique (4.5a)-(4.5d) réalisable. En effet, le fait de restreindre la longueur du pas d à l'aide d'une région de confiance peut empêcher ce dernier de satisfaire les contraintes linéarisées (4.6a)-(4.6b). Dès lors, le pas normal est limité à la région de confiance et a pour objectif de satisfaire *au mieux* les contraintes de (4.6a)-(4.6b), et cela au sens des moindres carrés. Ce pas est obtenu approximativement à l'aide d'une adaptation de la méthode dogleg appliquée au problème dérivé de la formulation du sous-problème suivant en la variable $v = (v_x, v_s)$:

$$\begin{array}{ll} min_v & ||A_h^T v_x + h||_2^2 + ||A_g^T v_x + v_s + g + s||_2^2 \\ s.c. : & ||(v_x, S^{-1} v_s)||_2 \leq \zeta \Delta \\ & v_s \geq -\frac{\tau s}{2}, \end{array} \tag{4.19}$$

où ζ est un paramètre choisi dans l'intervalle ouvert $(0, 1)$ et Δ est le rayon de la région de confiance. Effectuant le changement de variables

$$\tilde{v} = (v_x, \tilde{v}_s) = (v_x, S^{-1} v_s) \tag{4.20}$$

dans (4.19) et développant la fonction objectif tout en négligeant les termes constants, le sous-problème résolu par l'approche dogleg modifiée est donné par :

$$min_{\tilde{v}} \quad 2[\ h^T (g+s)^T\ ] \hat{A}^T \begin{bmatrix} v_x \\ \tilde{v}_s \end{bmatrix} + [\ v_x^T \quad \tilde{v}_s^T\ ] \hat{A} \hat{A}^T \begin{bmatrix} v_x \\ \tilde{v}_s \end{bmatrix} \tag{4.21a}$$

$$s.c. : \quad ||\tilde{v}||_2 \leq \zeta \Delta \tag{4.21b}$$

$$\tilde{v}_s \geq -\frac{\tau}{2}, \tag{4.21c}$$

où $\hat{A}$ est donnée par (4.16). La fonction objectif de (4.21a) sera par la suite appelée $m(\tilde{v})$ pour des raisons de simplicité.

La méthode commence par calculer le **point de Cauchy** (PC) pour le problème (4.21a)-(4.21c), en minimisant la fonction quadratique $m(\tilde{v})$ le long de la direction de la plus forte pente, à partir de $\tilde{v} = 0$. L'expression analytique de ce point est la suivante [3, page 886] :

$$\tilde{v}^{\text{PC}} = \begin{bmatrix} v_x^{\text{PC}} \\ \tilde{v}_s^{\text{PC}} \end{bmatrix} = -\alpha \hat{A} \begin{bmatrix} h \\ g+s \end{bmatrix}, \tag{4.22}$$

où

$$\alpha = \frac{\left\| \hat{A} \begin{bmatrix} h \\ g+s \end{bmatrix} \right\|_2^2}{\begin{bmatrix} h^T & g^T + s^T \end{bmatrix} (\hat{A}^T \hat{A})^2 \begin{bmatrix} h \\ g+s \end{bmatrix}}. \tag{4.23}$$

Ensuite, le **point de Newton** (N) est déterminé comme étant le minimiseur de norme minimale de $m(\tilde{v})$, obtenu par [3, page 886] :

$$\tilde{v}^{\text{N}} = \begin{bmatrix} v_x^{\text{N}} \\ \tilde{v}_s^{\text{N}} \end{bmatrix} = -\hat{A}(\hat{A}^T \hat{A})^{-1} \begin{bmatrix} h \\ g+s \end{bmatrix}. \tag{4.24}$$

Les pas de Cauchy et de Newton combinés constituent la **trajectoire dogleg** définie par deux segments linéaires, le premier partant de $\tilde{v} = 0$ jusque $\tilde{v} = \tilde{v}^{\text{PC}}$ et le second de $\tilde{v} = \tilde{v}^{\text{PC}}$ jusque $\tilde{v} = \tilde{v}^{\text{N}}$. La fonction $m(\tilde{v})$ est alors minimisée sous les contraintes (4.21b)-(4.21c), le long de cette trajectoire et le long de la direction de Newton (4.24), afin de calculer le pas $\tilde{v}$. Finalement, le pas v est obtenu en utilisant le changement de variables défini en (4.20). L'algorithme suivant reprend les différentes étapes de l'approche suivie pour déterminer le pas normal v.

Procédure Dogleg (DL)

Calculer $\tilde{v}^{\text{PC}}$ et $\tilde{v}^{\text{N}}$;

$\theta_1 = max\{\theta \in (0,1] \mid \theta\tilde{v}^{\text{N}}$ est réalisable$\}$;

Si $\theta_1 = 1$, *alors* :

$\tilde{v} = \tilde{v}^{\text{N}}$;

Sinon :

1. $\theta_2 = max\{\theta \in (0,1] \mid (1-\theta)\tilde{v}^{\text{PC}} + \theta\tilde{v}^{\text{N}}$ est réalisable pour $(4.21b) - (4.21c)\}$;

2. *si* une telle valeur de θ_2 n'existe pas, *alors* :
   $\theta_3 = max\{\theta \in (0,1] \mid \theta\tilde{v}^{\text{PC}} \text{ est réalisable}\}$;
   $\tilde{v}^{\text{DL}} = \theta_3\tilde{v}^{\text{PC}}$;

   sinon :
   $\tilde{v}^{\text{DL}} = (1-\theta_2)\tilde{v}^{\text{PC}} + +\theta_2\tilde{v}^{\text{N}}$;

   fin (si) ;

3. *si* $m(\tilde{v}^{\text{DL}}) < m(\theta_1\tilde{v}^{\text{N}})$, *alors* :
   $\tilde{v} = \tilde{v}^{\text{DL}}$;

   sinon :
   $\tilde{v} = \theta_1\tilde{v}^{\text{N}}$;

   fin (si) ;

Fin (Si) ;
$v = (v_x, S\tilde{v}_s)$.

Notons que comme le modèle m est convexe, il décroît le long de la trajectoire dogleg. De cette manière, le **point dogleg** $\tilde{v}^{\text{DL}}$ minimise m le long de cette trajectoire sous les contraintes (4.21b)-(4.21c).

4.4.2 Calcul du pas tangentiel

Dans un deuxième temps, la composante tangentielle de d, notée w, est déterminée. Pour cela, une fois le pas normal v calculé, les vecteurs r_h et r_g sont définis comme étant les résidus dans le calcul du pas normal :

$$r_h = A_h^T v_x + h \quad \text{et} \quad r_g = A_g^T v_x + v_s + g + s. \tag{4.25}$$

De cette manière, le sous-problème quadratique (4.5a)-(4.5d) prend la forme suivante :

$$\begin{aligned} min \quad & \nabla f^T d_x - \mu e^T S^{-1} d_s + \tfrac{1}{2}(d_x^T \nabla_{xx}^2 \mathcal{L} d_x + d_s^T \Sigma d_s) && \text{(4.26a)} \\ s.c. : \quad & A_h^T d_x = A_h^T v_x && \text{(4.26b)} \end{aligned}$$

$$A_g^T d_x + d_s = A_g^T v_x + v_s \tag{4.26c}$$
$$||(d_x, S^{-1}d_s)||_2 \leq \Delta \tag{4.26d}$$
$$d_s \geq -\tau s, \tag{4.26e}$$

où les dépendances en x^k, s^k, λ_h et λ_g ont été mises de côté afin d'alléger l'écriture. Notons que le choix des résidus r_h et r_g n'est pas innocent. En effet, si d est choisi comme étant le pas normal obtenu précédemment, les contraintes de (4.26a)-(4.26e) sont satisfaites et donc réalisables. D'autre part, ce choix est motivé par le fait que d est contraint de réaliser le plus de progrès possible, dans le but de satisfaire les contraintes, tout comme le pas v.

Afin de résoudre approximativement le sous-problème (4.26a)-(4.26e), le pas d est écrit comme une combinaison linéaire des pas v et w, i.e., $d = v + w$, où w est à déterminer. Comme son nom l'indique, w est défini comme un pas tangent au gradient des contraintes. Un changement de variables, identique à (4.20), est alors introduit de sorte que :

$$\tilde{d} = \begin{pmatrix} \tilde{d}_x \\ \tilde{d}_s \end{pmatrix} = \begin{pmatrix} d_x \\ S^{-1}d_s \end{pmatrix} = \begin{pmatrix} v_x \\ \tilde{v}_s \end{pmatrix} + \begin{pmatrix} w_x \\ \tilde{w}_s \end{pmatrix} = \tilde{v} + \tilde{w}. \tag{4.27}$$

A l'aide de cette relation, le sous-problème (4.26a)-(4.26e) est reformulé en la composante tangentielle $\tilde{w}$, en suivant les étapes ci-après, de manière à obtenir un sous-problème dit tangentiel. Ainsi, introduisant (4.27) dans (4.26a)-(4.26e), la fonction objectif de ce problème s'exprime de la manière suivante :

$$q(\tilde{v} + \tilde{w}) \equiv (\nabla f^T, -\mu e^T)(\tilde{v} + \tilde{w}) + \frac{1}{2}(\tilde{v} + \tilde{w})^T G(\tilde{v} + \tilde{w}), \tag{4.28}$$

où

$$G = \begin{pmatrix} \nabla^2_{xx}\mathcal{L} & 0 \\ 0 & S\Sigma S \end{pmatrix}, \tag{4.29}$$

et la contrainte (4.26d) devient :

$$||\tilde{d}||_2^2 = ||\tilde{v} + \tilde{w}||_2^2 \leq \Delta^2. \tag{4.30}$$

Cependant, rappelant la définition (4.16) de la matrice $\hat{A}$, nous savons que $\tilde{w}$ satisfait $\hat{A}^T\tilde{w} = 0$, i.e., $\tilde{w} \in Ker(\hat{A}^T)$, de même que $\tilde{v}$ se trouve dans l'espace $Im(\hat{A}^T)$. Il s'en suit donc que $\tilde{w}^T\tilde{v} = 0$. Dès lors, en développant la norme au carré dans la relation (4.30), la contrainte (4.26d) est finalement donnée par :

$$||\tilde{w}||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2. \tag{4.31}$$

Par conséquent, le sous-problème (4.26a)-(4.26e) est réécrit de la façon suivante :

$$
\begin{aligned}
min_{\tilde{w}} \quad & q(\tilde{v}) + \nabla f^T w_x - \mu e^T \tilde{w}_s + (G\tilde{v})^T \tilde{w} + \tfrac{1}{2}(\tilde{w}^T G \tilde{w}) && \text{(4.32a)}\\
s.c. : \quad & A_h^T w_x = 0 && \text{(4.32b)}\\
& A_g^T w_x + S\tilde{w}_s = 0 && \text{(4.32c)}\\
& ||\tilde{w}||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2 && \text{(4.32d)}\\
& \tilde{w}_s \geq -\tau e - \tilde{v}_s, && \text{(4.32e)}
\end{aligned}
$$

où la fonction objectif est désignée par $q(\tilde{v} + \tilde{w})$, pour des raisons de simplicité. Ce sous-problème est appelé **sous-problème tangentiel**. Dans le but de calculer une approximation du pas tangentiel w, la méthode du Gradient Conjugué (GC), proposée par Steihaug, est utilisée, méthode à laquelle sera appliqué un préconditionnement [3]. Plutôt que de présenter une simple itération GC, nous allons à présent décrire les étapes menant à une telle itération, en commençant par motiver la stratégie de préconditionnement.

Comme $\tilde{w} \in Ker(\hat{A}^T)$, ce pas peut s'exprimer à l'aide d'une base $\tilde{Z}$ de l'espace $Ker(\hat{A}^T)$, i.e. :

$$
\tilde{w} = \tilde{Z}u \equiv \begin{pmatrix} Z_x \\ \tilde{Z}_s \end{pmatrix} u, \tag{4.33}
$$

$\forall u \in \mathbb{R}^{n-t}$. Ainsi, tout pas de la forme (4.33) satisfait les contraintes (4.32b)-(4.32c), puisque ces dernières peuvent être formulées de manière équivalente par $\hat{A}^T \tilde{w} = 0$. Dès lors, le sous-problème tangentiel devient :

$$
\begin{aligned}
min_u \quad & q(\tilde{v} + \tilde{Z}u) && \text{(4.34a)}\\
s.c. : \quad & ||\tilde{Z}u||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2 && \text{(4.34b)}\\
& \tilde{Z}_s u \geq -\tau e - \tilde{v}_s. && \text{(4.34c)}
\end{aligned}
$$

Celui-ci est alors résolu en appliquant la méthode préconditionnée du Gradient Conjugué. Le préconditionnement d'une itération GC a pour objectif d'accélérer cette itération. En effet, si la méthode GC sans préconditionnement est appliquée pour minimiser (4.34a), un mauvais choix de la matrice $\tilde{Z}$ pourrait ralentir la progression de l'itération GC, car la matrice Hessienne de (4.34a) est :

$$
\tilde{Z}^T G \tilde{Z}.
$$

Dès lors, un mauvais choix de $\tilde{Z}$ pourrait rendre cette matrice mal conditionnée. Pour soulever cette difficulté, l'itération GC est préconditionnée par

la matrice :

$$\tilde{Z}^T\tilde{Z}, \tag{4.35}$$

dans quel cas le taux de convergence est gouverné par le spectre de :

$$(\tilde{Z}^T\tilde{Z})^{-\frac{1}{2}}\tilde{Z}^T G\tilde{Z}(\tilde{Z}^T\tilde{Z})^{-\frac{1}{2}}. \tag{4.36}$$

L'itération GC commence par calculer les estimateurs des minimiseurs de (4.34a) par la récurrence suivante :

$$u^+ = u + \alpha\delta, \tag{4.37}$$

où le paramètre α est obtenu en minimisant la fonction objectif quadratique q le long de la direction conjuguée δ. Soulignons que deux directions δ et δ^+ sont dites **conjuguées** par rapport à la matrice G, si elles satisfont la relation $\delta^T G\delta^+ = 0$. Ensuite, le gradient de q par rapport à u étant donné par $\tilde{Z}^T\nabla q(\tilde{v} + \tilde{Z}u)$ et puisque le préconditionneur est défini par (4.35), les directions conjuguées δ sont générées par :

$$\delta^+ = -(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T\nabla q(\tilde{v} + \tilde{Z}u^+) + \beta\delta, \tag{4.38}$$

où β est un paramètre initialisé à zéro et choisi ensuite afin de maintenir la conjugaison entre les directions δ. Cependant, de par le coût des calculs lors des manipulations dues au préconditionneur (4.35), il est préférable d'exécuter l'itération GC dans l'espace tout entier plutôt que dans l'espace réduit. Dès lors, revenant à la variable $\tilde{w}$ par la relation (4.33), l'itération GC pour le problème (4.34a)-(4.34c) est la suivante :

$$\tilde{w}^+ = \tilde{w} + \alpha p, \tag{4.39}$$

$$p^+ = -\tilde{Z}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T\nabla q(\tilde{v} + \tilde{w}^+) + \beta p, \tag{4.40}$$

où $p \equiv \tilde{Z}\delta$. Remarquons cependant que la matrice $\tilde{Z}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T$ n'est autre que la projection orthogonale sur l'espace $Ker(\hat{A}^T)$ et peut donc s'exprimer par :

$$P = \tilde{Z}(\tilde{Z}^T\tilde{Z})^{-1}\tilde{Z}^T = I - \hat{A}(\hat{A}^T\hat{A})^{-1}\hat{A}^T. \tag{4.41}$$

En raison de la contrainte imposée par la région de confiance et de l'éventuelle non-convexité du modèle quadratique, l'itération GC se termine si le gradient projeté de q est plus petit qu'une tolérance prescrite, si la direction p^+ est une direction de courbure négative, ou si les itérés engendrés se

trouvent en dehors de la région de confiance. Ainsi, la procédure GC Projetée (GCP), décrite au cours de cette section et permettant de résoudre le sous-problème (4.32a)-(4.32e), est résumée par l'algorithme GCP. Notons que si la contrainte de borne (4.32e) n'est pas satisfaite par le nouvel itéré $\tilde{w}^+$, une étape supplémentaire est effectuée à la sortie de la boucle. Celle-ci consiste à restaurer le dernier itéré admissible $\tilde{w}$ ainsi que la direction p calculée à ce point.

Toutefois, avant d'énoncer cet algorithme, nous allons développer certains calculs dans le but d'établir une expression analytique des paramètres α et β. Ce paragraphe s'avère donc un peu plus technique que les autres. Pour commencer, reprenons la définition de α, généralisation de celle donnée à la suite de la relation (4.37). Dans l'espace noyau $Ker(\hat{A}^T)$, ce paramètre est défini comme le pas minimisant la fonction objectif quadratique q du sous-problème (4.32a)-(4.32e), le long de la direction conjuguée p. Sa valeur est donc obtenue en résolvant un problème du type :

$$min_{\alpha \geq 0} \quad \Theta(\alpha) \equiv q(\tilde{v} + \tilde{w}^+), \tag{4.42}$$

où $\tilde{w}^+$ est défini par (4.39), ce qui revient à chercher un point stationnaire de la fonction Θ, en résolvant l'équation :

$$\begin{aligned} \Theta(\alpha)' &= 0 \\ \Leftrightarrow \nabla q(\tilde{v} + \tilde{w}^+)^T p &= 0. \end{aligned} \tag{4.43}$$

Sachant par (4.32a) que :

$$\begin{aligned} \nabla q(\tilde{v} + \tilde{w}^+) &= (\nabla f, -\mu e) + G\tilde{v} + G(\tilde{w} + \alpha p) \\ &= (\nabla f, -\mu e) + G(\tilde{v} + \tilde{w}) + \alpha G p, \end{aligned} \tag{4.44}$$

et posant :

$$\begin{aligned} r &= \nabla q(\tilde{v} + \tilde{w}) \\ &= (\nabla f, -\mu e) + G(\tilde{v} + \tilde{w}), \end{aligned} \tag{4.45}$$

l'équation (4.43) devient :

$$r^T p + \alpha p^T G p = 0, \tag{4.46}$$

relation à partir de laquelle nous pouvons extraire une première expression de α, à savoir :

$$\alpha = \frac{-r^T p}{p^T G p}. \tag{4.47}$$

En utilisant la définition des directions conjuguées p pour la méthode du GC, cette expression peut être améliorée. En effet, par (4.40), le produit $r^T p$ s'écrit également de la manière suivante :

$$r^T p = -r^T g + \beta r^T p^-, \tag{4.48}$$

où $g = Pr$ et p^- est la direction conjuguée précédant p. Cependant, au vu du fait que $r^T p^- = 0$ [7, page 525], il suit que :

$$r^T p = -r^T g, \tag{4.49}$$

et, par conséquent, en remplaçant dans (4.47), l'expression finale du pas α est donnée par :

$$\alpha = \frac{r^T g}{p^T G p}. \tag{4.50}$$

En ce qui concerne le paramètre β, celui-ci a pour objectif de maintenir la conjugaison par rapport à la matrice G, entre les directions p calculées au cours des itérations GC. Dès lors, par définition des directions conjuguées, β est tel que la relation :

$$p^T G p^+ = 0, \tag{4.51}$$

est vérifiée. Introduisant la définition (4.40) de p^+ dans (4.51), nous obtenons que :

$$-p^T G g^+ + \beta p^T G p = 0, \tag{4.52}$$

et donc, β est évalué à l'aide de la relation :

$$\beta = \frac{(g^+)^T G p}{p^T G p}. \tag{4.53}$$

Comme dans le raisonnement précédent, nous pouvons voir que cette expression peut être améliorée. En effet, vu que $r^+ = r + \alpha G p$ par (4.45), le numérateur de (4.53) devient :

$$\begin{aligned} (g^+)^T G p &= \frac{1}{\alpha}[(g^+)^T r^+ - (g^+)^T r] \\ &= \frac{(g^+)^T r^+}{\alpha}, \end{aligned} \tag{4.54}$$

où la dernière égalité provient du fait que $(Pr^+)^T r = 0$ [7, page 525], tandis que le dénominateur prend la forme :

$$\begin{aligned} p^T G p &= \frac{1}{\alpha}[p^T r^+ - p^T r] \\ &= \frac{-r^T p}{\alpha}, \end{aligned} \tag{4.55}$$

puisque $p^T r^+ = 0$. Néanmoins, il est encore possible de développer le produit $r^T p$ de la manière suivante :

$$\begin{aligned} r^T p &= -r^T g + \beta r^T p^- \\ &= -r^T g, \end{aligned} \tag{4.56}$$

en utilisant de nouveau le fait que $r^T p^- = 0$, de sorte que l'expression analytique finale de β est :

$$\beta = \frac{(r^+)^T g^+}{r^T g}. \tag{4.57}$$

A présent, nous avons en main tous les outils nécessaires afin de présenter l'algorithme GCP.

Algorithme GCP

Poser $\tilde{w} = 0$, $r = (r_x, r_s) = (\nabla f, -\mu e) + G\tilde{v}$, $g = Pr$, $p = -g$, $tol = 0.01\sqrt{g^T r}$;

Répéter au plus $2(n-t)$ *fois* ou *jusqu'à ce qu'un critère d'arrêt soit satisfait* :

1. *si* $p^T G p \leq 0$, *alors* :
   $\tilde{w}^+ = \tilde{w} + \theta p$, où $\theta > 0$ est tel que $||\tilde{w}^+||_2 = \Delta$;
   STOP ;

2. $\alpha = \frac{r^T g}{p^T G p}$;

3. $\tilde{w}^+ = \tilde{w} + \alpha p$;

4. *si* $||\tilde{w}^+||_2 > \Delta$, *alors* :
   $\tilde{w}^+ = \tilde{w} + \theta p$, où $\theta > 0$ est tel que $||\tilde{w}^+||_2 = \Delta$;

STOP;

5. $r^+ = r + \alpha Gp$;

6. $g^+ = Pr^+$;

7. *si* $(g^+)^T r^+ < tol$, *alors* :
   STOP;

8. $\beta = \frac{(r^+)^T g^+}{r^T g}$;

9. $p^+ = -g^+ + \beta p$;

10. $\tilde{w} \leftarrow \tilde{w}^+$, $r \leftarrow r^+$, $p \leftarrow p^+$;

Fin (boucle);
Si $\tilde{w}$ ne satisfait pas (4.32e), *alors* :

1. poser $\tilde{w}^+ = \tilde{w} + \theta p$, où $\tilde{w}$ est le dernier itéré admissible, p la direction correspondante et θ la plus grande valeur > 0 telle que $\tilde{w} + \theta p$ soit admissible;

2. poser $w = (w_x, w_s) = (\tilde{w}_x^+, S\tilde{w}_s^+)$;

Sinon :

1. Poser $w = (w_x, w_s) = (\tilde{w}_x, S\tilde{w}_s)$;

Fin (Si).

Cette procédure vient clôturer la description de la résolution du sous-problème quadratique (4.5a)-(4.5d) dont la solution d est le pas recherché dans l'algorithme PQS de région de confiance. Avant de reprendre l'algorithme complet décrivant la méthode de points intérieurs qui fait l'objet de ce chapitre, nous allons discuter, au cours des deux prochaines sections, quelques détails algorithmiques.

4.5 Fonction de mérite

La fonction de mérite ϕ, définie en (4.14), est utilisée, d'une part, afin de déterminer si le pas total $d = v + w$ est acceptable. D'autre part, elle fournit des informations concernant la mise à jour du rayon Δ de la région de confiance. Afin de la spécifier complètement, analysons le rôle du paramètre de pénalité ν introduit dans sa définition. L'introduction de ce paramètre a pour objectif d'établir un équilibre entre la contribution relative de la fonction objectif du sous-problème barrière (4.2) et celle de ses contraintes. A chaque itération, ν est choisi de telle sorte que d et ϕ soient compatibles. Dans le but de déterminer la manière dont ν est mis à jour, les variations de ϕ, dues au pas d, sont approximées par la **réduction prédite**, définie par :

$$pred(d) = -q(\tilde{v} + \tilde{w}) + \nu v_{pred}, \tag{4.58}$$

où q est la fonction objectif du sous-problème tangentiel, (4.32a), et où v_{pred} est la réduction de ϕ fournie par le pas normal et donnée selon la relation :

$$v_{pred} = \left\| \begin{bmatrix} h \\ g + s \end{bmatrix} \right\| - \left\| \begin{bmatrix} h \\ g + s \end{bmatrix} + \hat{A}^T \tilde{v} \right\|. \tag{4.59}$$

Le paramètre ν est alors contraint de prendre la plus grande valeur possible de telle sorte que la réduction $pred(d)$ soit positive et proportionnelle à v_{pred}, i.e. :

$$pred(d) \geq \rho \nu v_{pred}, \tag{4.60}$$

où $0 < \rho < 1$. Introduisant la définition (4.58) dans (4.60), il suit que ν doit être choisi de manière à vérifier :

$$\nu \geq \frac{q(\tilde{v} + \tilde{w})}{(1 - \rho) v_{pred}}. \tag{4.61}$$

Cependant, dans le cas où $m(\tilde{v}) = 0$, nous avons que $\tilde{v} = 0$, ce qui implique que $q(\tilde{v} + \tilde{w}) \leq 0$, [2]. Par conséquent, la condition (4.60) est satisfaite pour toute valeur positive de ν et, en particulier, pour la valeur qui lui était attribuée à l'itération précédente de l'algorithme PQS de région de confiance, i.e., pour ν^-. La mise à jour du paramètre de pénalité se résume donc à l'aide de la procédure suivante :

Algorithme de mise à jour du paramètre de pénalité

Si $m(\tilde{v}) = 0$, *alors* :

$$\nu = \nu^{-} ;$$

Sinon :

$$\nu = max\left\{\nu^{-}, \frac{q(\tilde{v}+\tilde{w})}{(1-\rho)v_{pred}}\right\} ;$$

Fin (Si).

Soulignons que cet algorithme s'applique pour une valeur fixée du paramètre barrière μ. Par conséquent, pour un sous-problème barrière fixé, ν augmente de façon monotone au cours des itérations. Cette propriété de croissance monotone est importante afin d'assurer la convergence globale de la méthode de points intérieurs décrite à travers ce chapitre.

A présent, nous pouvons considérer la manière dont la fonction de mérite entre en jeu pour déterminer si d est un pas acceptable pour l'algorithme PQS. Pour cela, la **réduction réelle** obtenue dans ϕ est calculée :

$$ared(d) = \phi(x, s; \nu) - \phi(x + d_x, s + d_s; \nu). \tag{4.62}$$

Le pas d est alors accepté si il entraîne une réduction suffisante dans la valeur de ϕ, dans le sens où l'inégalité suivante est vérifiée :

$$\gamma \equiv \frac{ared(d)}{pred(d)} \geq \eta, \tag{4.63}$$

où $0 < \eta < 1$. Soulignons que R.H. Byrd, M.E. Hribar et J. Nocedal ont utilisé la valeur $\eta = 10^{-8}$ dans les différents tests qu'ils ont effectués [3]. On peut voir que la relation (4.63) est satisfaite si le rayon de la région de confiance est suffisamment petit [3]. Ainsi, si d est accepté, le rayon Δ de la région de confiance est modifié comme suit :

$$\Delta^{+} = \begin{cases} max\{7||d||, \Delta\} & \text{si} \quad \gamma \geq 0.9, \\ max\{2||d||, \Delta\} & \text{si} \quad 0.3 \leq \gamma < 0.9, \\ \Delta & \text{si} \quad \eta \leq \gamma < 0.3. \end{cases} \tag{4.64}$$

D'autre part, si d est rejeté, Δ est réduit au plus de la moitié de la longueur du pas d, mais pas de moins du dixième de cette longueur. Notons que lorsque la valeur du paramètre barrière μ est diminuée, Δ est ajusté en utilisant la règle $\Delta \leftarrow max(5\Delta, 1)$.

Dans le but d'obtenir une convergence rapide de la méthode, la région de confiance se doit d'être inactive près de la solution, de sorte que l'algorithme de points intérieurs puisse alors prendre des pas de Newton complets. Toutefois, de par la non-différentiabilité de la fonction de mérite, il se peut qu'un pas qui approche le point solution ne satisfasse pas la condition (4.63) et soit donc rejeté. Ce phénomène est appelé effet de Maratos et se traduit par un accroissement de la norme des contraintes, dû à leur non-linéarité. Une manière de rectifier la situation est d'introduire un pas correctif du second ordre, noté y [3]. Ce pas représente en fait un pas de Newton sur les contraintes, calculé au point $x + d$ à partir de la relation (4.24). Dans la méthode que nous décrivons, cette correction n'a lieu que lorsque la composante normale de d est relativement petite par rapport à la composante tangentielle, ce qui est résumé dans la procédure de Correction du Second Ordre (CSO).

Algorithme CSO

Si $||\tilde{v}|| \leq 0.1||\tilde{w}||$, *alors* :

$$y = \hat{A}(\hat{A}^T\hat{A})^{-1}\begin{bmatrix} h(x+d_x) \\ g(x+d_x)+s+d_s \end{bmatrix};$$

Sinon :

$$y = 0;$$

Fin (Si).

Finalement, nous concluons cette section en précisant que, dans le cas où une correction du second ordre est nécessaire, le pas total recherché par l'algorithme PQS de région de confiance est donné par $d + y$.

4.6 Solution des systèmes linéaires

L'algorithme barrière décrit au cours des sections précédentes nécessite de résoudre trois systèmes linéaires par itération. Ceux-ci apparaissent dans le calcul des multiplicateurs de Lagrange (4.15), dans la composante de Newton du pas normal (4.24) et dans la projection Pr^+ recquise par la procédure GCP, où P est défini par (4.41). Ces systèmes peuvent être résolus en utilisant seulement une factorisation matricielle.

Le pas normal (4.24) demande la solution d'un système du type :

$$\hat{A}^T\hat{A}x = b,$$

où $\hat{A}$ est donnée en (4.16). Sa solution est alors calculée en résolvant le système augmenté suivant :

$$\begin{bmatrix} I & \hat{A} \\ \hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} z \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ -b \end{bmatrix}. \tag{4.65}$$

De manière similaire, le calcul de $g = Pr$, où P est exprimé en termes de $\hat{A}$, (4.41), peut être exécuté en résolvant :

$$\begin{bmatrix} I & \hat{A} \\ \hat{A}^T & 0 \end{bmatrix} \begin{bmatrix} g \\ l \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}. \tag{4.66}$$

De plus, si r est remplacé dans (4.66) par $(-\nabla f, \mu e)^T$, le vecteur l contient alors, par (4.15), les multiplicateurs de Lagrange au sens des moindres carrés. Notons que, selon R.H. Byrd, M.E. Hribar et J. Nocedal, il semble plus exact de travailler avec le système augmenté (4.65), plutôt que de factoriser la matrice d'équations normales $\hat{A}^T\hat{A}$, [3].

4.7 Description complète de la méthode barrière de points intérieurs

Ayant détaillé les différentes étapes de la méthode de points intérieurs définie par R.H. Byrd, M.E. Hribar et J. Nocedal, nous pouvons à présent donner l'algorithme complet permettant de résoudre des problèmes de programmation non linéaire de grande taille, de la forme (4.1). Cet algorithme, dont le nom KNITRO est un acronyme pour « Nonlinear Interior point Trust Region Optimizer », implémente une méthode de points intérieurs primale ou primale-duale, selon la définition de Σ_k et des multiplicateurs de Lagrange λ_k, [Section 4.3]. Avant d'énoncer ce dernier, rappelons que les conditions d'arrêt pour chaque sous-problème barrière, ainsi que pour l'algorithme entier, sont basées sur la fonction $E(x, s; \mu)$, définie en (4.4), où $(\lambda_h, \lambda_g) = \lambda^{\text{LS}}(x, s; \mu)$ est donné en (4.15).

Algorithme KNITRO

Choisir une valeur pour les paramètres $\eta > 0$, $\tau \in (0,1)$, $\theta \in (0,1)$ et $\zeta \in (0,1)$;

Sélectionner les tolérances d'arrêt ϵ_μ et ϵ_{TOL} ;

Choisir une valeur initiale pour μ, x^0, $s^0 > 0$ et Δ_0, et poser $k = 0$;

Répéter jusqu'à ce que $E(x^k, s^k; \mu) \leq \epsilon_{\text{TOL}}$:

Répéter jusqu'à ce que $E(x^k, s^k; \mu) \leq \epsilon_\mu$:

1. calculer le pas normal $v_k = (v_x, v_s)$ par la Procédure Dogleg, décrite à la Section 4.4.1 ;

2. calculer les multiplicateurs de Lagrange à partir de (4.18) ;

3. calculer $\nabla^2_{xx}\mathcal{L}(x^k, s^k, \lambda_h, \lambda_g)$ et Σ_k en utilisant (4.9) ou (4.13) ;

4. calculer le pas tangentiel $w_k = (w_x, w_s)$ par l'Algorithme GCP, défini dans la Section 4.4.2 ;

5. calculer le pas total $d_k = v_k + w_k$;

6. mettre à jour ν_k par l'Algorithme de mise à jour du paramètre de pénalité, Section 4.5 ;

7. calculer $pred_k(d_k)$ par (4.58) et $ared_k(d_k)$ par (4.62) ;

8. *si* $ared_k(d_k) \geq \eta pred_k(d_k)$, *alors* :
   poser $x^{k+1} = x^k + d_x$;
   poser $s^{k+1} = s^k + d_s$;
   mettre à jour Δ_{k+1} par (4.64) ;

   sinon exécuter l'Algorithme CSO, Section 4.5, pour obtenir $y_k = (y_x, y_s)$:
   si $y_k \neq 0$, *si* $ared_k(d_k + y_k) \geq \eta pred_k(d_k)$
   et *si* $s^k + d_s + y_s \geq (1 - \tau)s^k$, *alors* :

poser $x^{k+1} = x^k + d_x + y_x$;

poser $s^{k+1} = s^k + d_s + y_s$;

poser $\Delta_{k+1} = \Delta_k$;

sinon :

poser $x^{k+1} = x^k$;

poser $s^{k+1} = s^k$;

poser $\Delta_{k+1} \in [0.1\Delta_k, 0.5\Delta_k]$;

fin (si) ;

fin (si) ;

9. poser $k \leftarrow k + 1$;

Fin (boucle) ;

$\mu \leftarrow \theta\mu$, $\epsilon_\mu \leftarrow \theta\epsilon_\mu$;

Restaurer ν_{k-1} et Δ_k ;

Fin (boucle).

La convergence globale de l'agorithme KNITRO a été prouvée par R.H. Byrd, J.C. Gilbert et J. Nocedal, [2], tandis que la convergence superlinéaire locale a été démontrée dans [4], en utilisant un taux de décroissance superlinéaire pour le paramètre barrière μ.

Cet algorithme clôture la description d'une méthode de points intérieurs en programmation non linéaire. Notons qu'à travers ce chapitre, nous ne nous sommes pas préoccupés de la manière dont le point de départ est calculé. C'est pourquoi, au cours du chapitre suivant, nous décrivons un procédé permettant de calculer un point de départ pour des algorithmes en programmation non linéaire.

Chapitre 5

Calcul de Points de Départ en Programmation Non Linéaire

5.1 Introduction

Au cours de ce chapitre, une heuristique de calcul du point de départ pour des algorithmes de points intérieurs est présentée. Cette nouvelle stratégie est en cours de développement et les mathématiciens contribuant à celle-ci sont M. Gertz, J. Nocedal et A. Sartenaer [6].

Avant d'expliquer la stratégie de calcul du point de départ proprement dite, posons le cadre de travail auquel celle-ci est appliquée. Pour cela, le problème de programmation non linéaire (4.1), introduit au chapitre précédent, est repris :

$$\begin{array}{ll} min & f(x) \\ s.c. : & h(x) = 0 \\ & g(x) \leq 0, \end{array} \tag{5.1}$$

où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $h : \mathbb{R}^n \longrightarrow \mathbb{R}^t$ et $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ sont des fonctions deux fois continuement différentiables, ainsi que le Lagrangien qui lui est associé :

$$\mathcal{L}(x, \lambda_h, \lambda_g) \;=\; f(x) + \lambda_h^T h(x) + \lambda_g^T g(x). \tag{5.2}$$

En introduisant une variable d'écart s, dans le problème (5.1), celui-ci peut également s'écrire :

$$\begin{array}{ll} min & f(x) \\ s.c. : & h(x) = 0 \end{array} \tag{5.3}$$

$$g(x) + s = 0$$
$$s \geq 0,$$

où f, h et g sont définies de manière identique que pour (5.1), tandis que, dans ce cas, le Lagrangien associé est de la forme :

$$\mathcal{L}(x, \lambda_h, \lambda_g, s, \lambda_s) = f(x) + \lambda_h^T h(x) + \lambda_g^T (g(x) + s) - \lambda_s^T s. \quad (5.4)$$

Dès lors, les conditions d'optimalité du premier ordre du problème (5.3) sont données par le système :

$$\begin{aligned} r(x, \lambda_h, \lambda_g, s, \lambda_s) &= 0, & (5.5a)\\ s^T \lambda_s &= 0, & (5.5b)\\ s, \lambda_s &\geq 0, & (5.5c) \end{aligned}$$

où r est une fonction vectorielle dont les composantes sont définies par :

$$\begin{aligned} r_x(x, \lambda_h, \lambda_g) &= \nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g, & (5.6a)\\ r_{\lambda_h}(x) &= h(x), & (5.6b)\\ r_s(x, s) &= g(x) + s, & (5.6c)\\ r_{\lambda_g}(\lambda_g, \lambda_s) &= \lambda_g - \lambda_s, & (5.6d) \end{aligned}$$

avec, respectivement, $A_h^T(x)$ et $A_g^T(x)$ les matrices Jacobiennes des fonctions h et g, déjà introduites au chapitre précédent [Chapitre 4, Section 4.2]. Plutôt que d'utiliser l'équation (5.6d) pour éliminer λ_s, comme c'est souvent le cas pour les méthodes de points intérieurs, nous garderons cette équation afin de maintenir des valeurs séparées pour λ_g et λ_s. De cette manière, une plus grande flexibilité apparaît dans le choix de la valeur initiale de λ_g. Nous reviendrons sur ce point dans la section suivante.

Partant d'un itéré $(x^k, \lambda_h^k, \lambda_g^k, s^k, \lambda_s^k)$, une méthode primale-duale de points intérieurs calcule une direction de recherche en appliquant la méthode de Newton au système suivant [Chapitre 1] :

$$\begin{aligned} r(x, \lambda_h, \lambda_g, s, \lambda_s) &= 0, & (5.7a)\\ S\lambda_s &= \mu e, & (5.7b) \end{aligned}$$

où $\mu > 0$ est le paramètre barrière, $S = diag(s_1, ..., s_m)$ et e désigne, dans ce cas, le vecteur unité dans $\mathbb{R}^m$. Ainsi, une direction de recherche, produite par une itération de Newton, est obtenue en résolvant :

$$\begin{pmatrix} H(x,\lambda_h,\lambda_g) & A_h(x) & A_g(x) & 0 & 0 \\ A_h(x)^T & 0 & 0 & 0 & 0 \\ A_g(x)^T & 0 & 0 & I & 0 \\ 0 & 0 & I & 0 & -I \\ 0 & 0 & 0 & \Lambda_s & S \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda_h \\ \Delta\lambda_g \\ \Delta s \\ \Delta\lambda_s \end{pmatrix}$$
$$= \begin{pmatrix} -\nabla_x\mathcal{L} \\ -h(x) \\ -g(x)-s \\ -\lambda_g+\lambda_s \\ -\Lambda_s s+\mu e \end{pmatrix}, \tag{5.8}$$

où

$$\begin{aligned} \nabla_x\mathcal{L} &= \nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g, \\ H(x,\lambda_h,\lambda_g) &= \nabla^2_{xx}\mathcal{L}(x,\lambda_h,\lambda_g), \\ \Lambda_s &= diag([\lambda_s]_1,...,[\lambda_s]_m). \end{aligned}$$

Une fois cette direction déterminée, le nouvel itéré est calculé en utilisant la relation suivante :

$$\begin{aligned} (x^{k+1},\lambda_h^{k+1},\lambda_g^{k+1},s^{k+1},\lambda_s^{k+1}) &= (x^k,\lambda_h^k,\lambda_g^k,s^k,\lambda_s^k) \\ &+\alpha(\Delta x^k,\Delta\lambda_h^k,\Delta\lambda_g^k,\Delta s^k,\Delta\lambda_s^k), \end{aligned} \tag{5.9}$$

où $\alpha > 0$ est la longueur de pas qui assure la décroissance d'une fonction de mérite et la positivité des variables s et λ_s [Chapitre 4]. Soulignons que la description de ce cadre de travail est plutôt simpliste car nous désirons présenter l'heuristique de calcul du point de départ de manière générale. Nous supposerons par la suite les itérés calculés à l'aide d'une itération de la forme (5.8)-(5.9).

Remarque : Comme ce fut le cas au chapitre précédent, nous utiliserons de nouveau la notation $[..]_i$ afin de désigner la $i^{\text{ème}}$ d'un vecteur déjà indicé, ainsi que la $i^{\text{ème}}$ composante d'un vecteur direction. Soulignons que cette notation sera également utilisée à travers le chapitre suivant.

5.2 Stratégie de calcul de points de départ

L'objectif de cette stratégie est de générer une point de départ $(x^0,\lambda_h^0,\lambda_g^0,s^0,\lambda_s^0)$, noté v^0 pour des raisons de simplicité, pour lequel le pas α effectué le

long de la direction primale-duale, issue de ce point et obtenue par résolution de (5.8), n'est pas trop petit. En effet, en raison de la contrainte de positivité imposée à s et λ_s, un pas complet peut ne pas toujours être effectué à partir de l'itéré courant. Néanmoins, il est désirable d'avancer le plus loin possible le long de cette direction, tout en satisfaisant les contraintes de positivité imposées à s et λ_s.

L'heuristique avancée par M. Gertz, J. Nocedal et A. Sartenaer commence par fixer des valeurs préliminaires $\tilde{x}$, $\tilde{\lambda}_h$, $\tilde{\lambda}_g$, $\tilde{s}$ et $\tilde{\lambda}_s$ de la manière suivante. D'une part, l'estimateur initial de la solution, $\tilde{x}$, est soit fournit par l'utilisateur, soit posé par défaut (par exemple, $\tilde{x} = 0$). D'autre part, les multiplicateurs de Lagrange $\tilde{\lambda}_h$ et $\tilde{\lambda}_g$ sont supposés calculés par la méthode de points intérieurs, tandis que les vecteurs $\tilde{s}$ et $\tilde{\lambda}_s$ sont fixés à une valeur constante strictement positive, notée δ. Une fois ces valeurs déterminées, la procédure attribue, de manière respective, les valeurs initiales $\tilde{x}$ et $\tilde{\lambda}_h$ aux composantes x^0 et λ_h^0 du point de départ, puisqu'aucune contrainte n'est imposée à ces variables. Ensuite, une direction affine (A), $\Delta^{\mathrm{A}} v = (\Delta^{\mathrm{A}} x, \Delta^{\mathrm{A}} \lambda_h, \Delta^{\mathrm{A}} \lambda_g, \Delta^{\mathrm{A}} s, \Delta^{\mathrm{A}} \lambda_s)$, est calculée en posant le paramètre μ à zéro dans le système primal-dual (5.8). Moyennant la connaissance de ce pas, deux vecteurs u et v de $\mathbb{R}^m$ sont évalués composante par composante, de sorte que :

$$u_i = max\{0, -(\tilde{s}_i + [\Delta^{\mathrm{A}} s]_i)\}, \tag{5.10}$$

$$v_i = max\{0, -([\tilde{\lambda}_s]_i + [\Delta^{\mathrm{A}} \lambda_s]_i)\}, \tag{5.11}$$

pour $i = 1, ..., m$. Le vecteur u représente la violation des contraintes de non-négativité primales, engendrée par la direction affine complète $\Delta^{\mathrm{A}} v$, tandis que v représente la violation correspondante des contraintes de non-négativité duales. Un scalaire ξ est alors utilisé afin de désigner la violation maximale de non-négativité. Celui-ci est défini par :

$$\xi = max\{u_i, v_i, i = 1, ..., m\}. \tag{5.12}$$

Ainsi, étant donné deux scalaires $\beta_1 > 0$ et $\beta_2 \geq 1$, les composantes initiales s^0 et λ_s^0 sont calculées en utilisant une des quatre règles suivantes :

R1 : $s_i^0 = max(\beta_1, \tilde{s}_i + [\Delta^{\mathrm{A}} s]_i)$ $[\lambda_s^0]_i = max(\beta_1, [\tilde{\lambda}_s]_i + [\Delta^{\mathrm{A}} \lambda_s]_i)$,

R2 : $s_i^0 = max(\beta_1, \tilde{s}_i + [\Delta^{\mathrm{A}} s]_i, u_i)$ $[\lambda_s^0]_i = max(\beta_1, [\tilde{\lambda}_s]_i + [\Delta^{\mathrm{A}} \lambda_s]_i, v_i)$,

R3 : $s^0 = \tilde{s} + \Delta^{\mathrm{A}} s + \beta_1 e + \beta_2 u$ $\lambda_s^0 = \tilde{\lambda}_s + \Delta^{\mathrm{A}} \lambda_s + \beta_1 e + \beta_2 v$,

R4 : $s^0 = \tilde{s} + \Delta^{\mathrm{A}} s + \beta_1 e + \beta_2 \xi e$ $\lambda_s^0 = \tilde{\lambda}_s + \Delta^{\mathrm{A}} \lambda_s + \beta_1 e + \beta_2 \xi e$,

où e est le vecteur unité dans $\mathbb{R}^m$. Lorsque les valeurs de ces composantes du point de départ sont déterminées, la procédure se poursuit en évaluant la valeur de λ_g^0. Pour cela, deux cas se distinguent. Le premier correspond à celui où les valeurs de λ_g et λ_s ne sont pas séparées, i.e., lorsque l'équation (5.6d) n'est pas maintenue. Dans ce cas, la composante λ_g^0 est obtenue en utilisant une des règles ci-dessus, où $\tilde{\lambda}_g$ joue le rôle de $\tilde{\lambda}_s$ et λ_g^0 celui de λ_s^0. Le second cas se rencontre lorsque l'équation (5.6d) est maintenue, et donc, quand les valeurs de λ_g et λ_s sont séparées. La valeur attribuée à λ_g^0 est alors la valeur préliminaire $\tilde{\lambda}_g$. Finalement, la stratégie se termine en mettant à jour la matrice Hessienne $H(x, \lambda_h, \lambda_g)$ au point de départ v^0, en utilisant la valeur de $\tilde{\lambda}_g$, i.e. :

$$H_0 \;=\; \tilde{H}, \tag{5.13}$$

où $\tilde{H} = H(\tilde{x}, \tilde{\lambda}_h, \tilde{\lambda}_g)$. La première raison qui motive l'utilisation de $\tilde{\lambda}_g$, plutôt que celle de λ_g^0, lors de cette mise à jour, est la suivante. Selon M. Gertz, J. Nocedal et A. Sartenaer [6], une examination précise des composantes de r, (5.6a)-(5.6d), montre que si H_0 ne dépend pas de la variable λ_g^0, alors ce Hessien ne dépend pas non plus du pas $\Delta v^0 = (\Delta x^0, \Delta\lambda_h^0, \Delta\lambda_g^0, \Delta s^0, \Delta\lambda_s^0)$. Dès lors, en utilisant (5.13), tous les algorithmes calculeront les mêmes directions de recherche, que les variables λ_g et λ_s soient considérées à valeurs séparées ou non. La deuxième motivation vient du fait que λ_g^0 peut prendre de grandes valeurs, introduisant alors une déformation indésirable dans le modèle quadratique utilisé par la méthode de Newton. En particulier, si une de ses composantes, notée $[\lambda_g^0]_i$, est très grande et que le terme du Hessien correspondant $\nabla^2 g_i(x^0)$ est indéfini, H_0 peut également devenir indéfini ce qui ralentirait l'itération. Enfin, en choisissant $H_0 = \tilde{H}$, une évaluation du Hessien $H(x, \lambda_h, \lambda_g)$ est évitée, empêchant ainsi un coût supplémentaire. Cette justification de la mise à jour (5.13) clôturant la description de l'heuristique de calcul du point de départ d'un programme non linéaire, nous pouvons présenter l'algorithme résumant cette stratégie.

Algorithme de calcul du point de départ en Programmation Non Linéaire

Choisir $\delta > 0$, $\beta_1 > 0$ et $\beta_2 \geq 1$;

Déterminer les valeurs préliminaires suivantes :

1. soit $\tilde{x}$ est fourni par l'utilisateur ;
   soit $\tilde{x}$ est fixé par défaut, par exemple $\tilde{x} = 0$;

2. calculer les multiplicateurs de Lagrange $\tilde{\lambda}_h$ et $\tilde{\lambda}_g$;

3. poser $\tilde{s} = \delta e$ et $\tilde{\lambda}_s = \delta e$;

Evaluer les fonctions f, h et g ainsi que leurs dérivées, au point $\tilde{x}$;
Calculer le direction affine $\Delta^A v$ en résolvant (5.8) avec $\mu = 0$;
Poser $(x^0, \lambda_h^0) \leftarrow (\tilde{x}, \tilde{\lambda}_h)$;
Si la méthode maintient des valeurs séparées pour λ_g et λ_s, *alors* :
$\lambda_g^0 \leftarrow \tilde{\lambda}_g$;

Sinon :
$\lambda_g^0 \leftarrow \lambda_s^0$;

Fin (Si) ;
Choisir s^0 et λ_s^0 en utilisant une des règles énoncées plus haut ;
Définir la matrice Hessienne initiale $H_0 = \tilde{H}$;
Débuter l'algorithme de points intérieurs à partir de v^0.

Au cours du prochain chapitre, cet algorithme est appliqué au cadre de la programmation linéaire, dans le but de l'utiliser comme méthode de calcul du point de départ de l'algorithme Prédicteur-Correcteur de Mehrotra (MPC) vu au premier chapitre. Le choix de cette application se justifie de la manière suivante. Les algorithmes de programmation non linéaire permettant de résoudre des problèmes plus compliqués que les algorithmes de programmation linéaire, il est toutefois pertinent de voir si ils résolvent de manière efficace des problèmes plus simples. En effet, il n'y aurait aucun intérêt à utiliser une méthode pour résoudre des problèmes plutôt compliqués, si cette méthode ne permet pas de résoudre efficacement des problèmes plus simples. C'est pourquoi nous avons analysé au cours du chapitre suivant l'application de l'heuristique de M. Gertz, J. Nocedal et A. Sartenaer, au départ d'un algorithme appartenant à la classe des méthodes de programmation linéaire. Nous avons sélectionné comme méthode l'algorithme MPC, et cela dans le but de comparer l'efficacité de l'heuristique avec chacune des trois méthodes de calcul du point de départ, vues au deuxième chapitre.

Chapitre 6

Application à la Programmation Linéaire

6.1 Introduction

Afin de tester l'heuristique présentée au chapitre précédent, nous avons décidé de l'appliquer au cadre de travail de la programmation linéaire [Chapitre 1, Section 1.3.3]. Pour cela, rappelons qu'un problème linéaire sous forme standard s'écrit :

$$\begin{array}{ll} min & c^T x \\ s.c.: & Ax = b \\ & x \geq 0, \end{array} \tag{6.1}$$

où $c, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times n}$, de rang m, et que les conditions d'optimalité, qui lui sont associées, sont données par le système (1.6a)-(1.6d). Dans les tests que nous avons effectués et qui sont présentés à la fin de cette section, l'heuristique calcule, en particulier, un point de départ pour l'algorithme MPC de Mehrotra [Section 1.6]. De cette manière, les résultats obtenus pour résoudre une gamme de problèmes, en démarrant la méthode de points intérieurs de la sorte, seront comparés aux résultats présentés au troisième chapitre, afin d'analyser les performances (ou non) de la stratégie proposée par M. Gertz, J. Nocedal et A. Sartenaer.

6.2 Approche de Gertz, Nocedal et Sartenaer

Avant de décrire l'application de cette approche au cadre linéaire, établissons, à l'aide du tableau suivant, un lien entre les variables utilisées en

programmation non linéaire (PNL), dans le Chapitre 5, et les variables correspondantes en programmation linéaire (PL), la troisième colonne précisant le rôle de ces variables. Notons que les variables de la PNL n'ayant aucune correspondance en PL n'ont pas été reprises dans ce tableau.

PNL	PL	Rôle des variables
s	x	variable soumise à la contrainte de positivité
λ_h	λ	multiplicateur de Lagrange associé à la contrainte d'égalité $h(x) = 0$ en PNL, resp. $b - Ax = 0$ en PL
λ_s	s	multiplicateur de Lagrange associé à la contrainte d'inégalité $g(x) \leq 0$ en PNL, resp. $x \geq 0$ en PL

TAB. 6.1 – Lien entre les variables de PNL et de PL.

Le rôle des variables nécessaires étant à présent fixé, nous pouvons adapter la stratégie vue au Chapitre 5, au calcul du point de départ (x^0, λ^0, s^0) d'une méthode de points intérieurs en PL. Tout d'abord, fixant les valeurs initiales $\tilde{x}$ et $\tilde{s}$ à une constante strictement positive, δ, et supposant le multiplicateur $\tilde{\lambda}$ évalué par la méthode de points intérieurs, la procédure calcule une direction affine $\Delta^{\mathrm{A}} v = (\Delta^{\mathrm{A}} x, \Delta^{\mathrm{A}} \lambda, \Delta^{\mathrm{A}} s)$ en résolvant le système (1.15), défini au premier chapitre. Ensuite, les vecteurs u et v de $\mathbb{R}^n$ sont déterminés par les relations suivantes :

$$u_i = max\{0, -(\tilde{x}_i + [\Delta^{\mathrm{A}} x]_i\}, \tag{6.2}$$

$$v_i = max\{0, -(\tilde{s}_i + [\Delta^{\mathrm{A}} s]_i\}, \tag{6.3}$$

pour $i = 1, ..., n$, vecteurs représentant respectivement la violation des contraintes de positivité primales et duales. Un scalaire ξ est également défini afin de déterminer la violation maximale des contraintes de positivité :

$$\xi = max\{u_i, v_i, i = 1, ..., n\}. \tag{6.4}$$

Moyennant alors la connaissance de $\beta_1 > 0$ et $\beta_2 \geq 1$, les points x^0 et s^0 sont obtenus en suivant une des quatre règles ci-dessous :

R1 : $\quad x_i^0 = max(\beta_1, \tilde{x}_i + [\Delta^{\mathrm{A}} x]_i) \qquad s_i^0 = max(\beta_1, \tilde{s}_i + [\Delta^{\mathrm{A}} s]_i),$

$$
\begin{array}{lll}
\text{R2} : & x_i^0 = max(\beta_1, \tilde{x}_i + [\Delta^{\mathrm{A}} x]_i, u_i) & s_i^0 = max(\beta_1, \tilde{s}_i + [\Delta^{\mathrm{A}} s]_i, v_i), \\
\text{R3} : & x^0 = \tilde{x} + \Delta^{\mathrm{A}} x + \beta_1 e + \beta_2 u & s^0 = \tilde{s} + \Delta^{\mathrm{A}} s + \beta_1 e + \beta_2 v, \\
\text{R4} : & x^0 = \tilde{x} + \Delta^{\mathrm{A}} x + \beta_1 e + \beta_2 \xi e & s^0 = \tilde{s} + \Delta^{\mathrm{A}} s + \beta_1 e + \beta_2 \xi e,
\end{array}
$$

où e est le vecteur unité dans $\mathbb{R}^n$, tandis que la composante λ^0 du point de départ est fixée à la valeur initiale $\tilde{\lambda}$, clôturant ainsi la description de la stratégie dans le cas linéaire. En effet, le Hessien du Lagrangien associé au problème (6.1) n'intervenant pas dans le calcul de la direction de recherche $\Delta^{\mathrm{A}} v$, la phase de mise à jour de celui-ci n'entre pas en compte. Par conséquent, nous pouvons résumer l'heuristique de calcul du point de départ pour des méthodes de points intérieurs en PL, à l'aide de l'algorithme suivant.

Algorithme de calcul du point de départ en programmation linéaire

Choisir $\delta > 0$, $\beta_1 > 0$ et $\beta_2 \geq 1$;

Déterminer les valeurs préliminaires suivantes :

1. poser $\tilde{x} = \delta e$ et $\tilde{s} = \delta e$;

2. calculer le multiplicateur de Lagrange $\tilde{\lambda}$;

Calculer le direction affine $\Delta^{\mathrm{A}} v$ en résolvant (1.15) ;

Poser $\lambda^0 \leftarrow \tilde{\lambda}$;

Choisir x^0 et s^0 en utilisant une des règles énoncées plus haut ;

Débuter l'algorithme de points intérieurs à partir de (x^0, λ^0, s^0).

Remarque : Au cours des tests qui suivront, les paramètres δ, β_1 et β_2 prennent respectivement les valeurs 1, 1000 et 2, suggérées par M. Gertz et ses confrères, et utilisées au cours des tests qu'ils ont effectués [6].

Nous avons vu au cours du premier chapitre que la seule condition imposée au point de départ de l'algorithme MPC est de satisfaire les contraintes de positivité en x et s, i.e., $(x^0, s^0) > 0$. Dès lors, terminons la description effectuée au cours de cette section en montrant que le point (x^0, λ^0, s^0), généré par l'algorithme ci-dessus, se trouve bien à l'intérieur de l'orthant de

non-négativité défini par $(x, s) \geq 0$, et cela quelle que soit la règle de calcul utilisée pour déterminer x^0 et s^0. Pour la première règle, il est évident que les vecteurs x^0 et s^0 calculés sont strictement positifs en toutes leurs composantes, puisque ces dernières prennent au minimum la valeur $\beta_1 > 0$. Par définition des vecteurs u et v et comme β_1 est strictement positif, il suit que la deuxième règle produit aussi deux points x^0 et s^0 strictement positifs. Finalement, en ce qui concerne les points obtenus par la troisième ou quatrième règle de calcul, ceux-ci vérifient également la condition de positivité. En effet, sachant que $\beta_2 \geq 1$, nous avons que u satisfait, par définition, les relations suivantes, pour $i = 1, ..., n$:

$$\begin{aligned} u_i \neq 0 \quad &\Rightarrow \quad \beta_2 u_i = \beta_2 |\tilde{x}_i + [\Delta^{\mathrm{A}} x]_i| > 0 \qquad (6.5) \\ &\qquad \text{et } \beta_2 u_i \geq \tilde{x}_i + [\Delta^{\mathrm{A}} x]_i, \\ u_i = 0 \quad &\Rightarrow \quad \tilde{x}_i + [\Delta^{\mathrm{A}} x]_i \geq 0, \qquad (6.6) \end{aligned}$$

de même que v vérifie les implications ci-dessous :

$$\begin{aligned} v_j \neq 0 \quad &\Rightarrow \quad \beta_2 v_j = \beta_2 |\tilde{s}_j + [\Delta^{\mathrm{A}} s]_j| > 0 \qquad (6.7) \\ &\qquad \text{et } \beta_2 v_j \geq \tilde{s}_j + [\Delta^{\mathrm{A}} s]_j, \\ v_j = 0 \quad &\Rightarrow \quad \tilde{s}_j + [\Delta^{\mathrm{A}} s]_j \geq 0, \qquad (6.8) \end{aligned}$$

pour $j = 1, ..., n$. De plus, puisque nous savons que $\beta_1 e > 0$, il suit que les vecteurs x^0 et s^0, calculé en utilisant la troisième règle, appartiennent à l'orthant de non-négativité défini par $(x, s) \geq 0$. D'autre part, comme ξ est, par définition, un scalaire positif ou nul, nous pouvons voir, par les relations (6.6) et (6.8), que :

$$\begin{aligned} \xi = 0 \quad &\Rightarrow \quad \tilde{x} + \Delta^{\mathrm{A}} x \geq 0 \qquad (6.9) \\ &\qquad \text{et} \quad \tilde{s} + \Delta^{\mathrm{A}} s \geq 0, \qquad (6.10) \end{aligned}$$

et, par la définition (6.4), que :

$$\begin{aligned} \xi > 0 \quad &\Rightarrow \quad \xi > \tilde{x}_i + [\Delta^{\mathrm{A}} x]_i \qquad (6.11) \\ &\qquad \text{et} \quad \xi > \tilde{s}_j + [\Delta^{\mathrm{A}} s]_j, \qquad (6.12) \end{aligned}$$

$\forall i = 1, ..., n$ tel que $\tilde{x}_i + [\Delta^{\mathrm{A}} x]_i < 0$ et $\forall j = 1, ..., n$ tel que $\tilde{s}_j + [\Delta^{\mathrm{A}} s]_j < 0$. Dès lors, par le fait que $\beta_1 e > 0$, il suit que la quatrième règle de calcul permet également d'engendrer deux vecteurs x^0 et s^0 strictement positifs.

6.3 Comparaison avec les trois approches décrites en Programmation Linéaire

Comparons à présent la méthode de calcul du point de départ, présentée au cours de la section précédente, avec les trois stratégies vues dans le cadre de la programmation linéaire. Tout comme celles-ci, l'approche de M. Gertz, J. Nocedal et A. Sartenaer détermine un point de départ (x^0, λ^0, s^0) strictement positif, en commençant par calculer un point préliminaire, noté $(\tilde{x}, \tilde{\lambda}, \tilde{s})$. Contrairement à ce qui se passait pour les méthodes utilisées en programmation linéaire, le point $(\tilde{x}, \tilde{\lambda}, \tilde{s})$ ne satisfait pas les contraintes d'égalité du problème (6.1) et de son dual, donné par (1.2), mais se trouve à l'intérieur de l'orthant de non-négativité défini par la relation $(x, s) \geq 0$. Ainsi, le tableau suivant reprend, colonne par colonne, les caractéristiques de cette approche, caractéristiques que nous avions analysées pour les trois autres méthodes [TAB. 2.1].

Approche de Gertz et ses confrères			
$(\tilde{x}, \tilde{\lambda}, \tilde{s})$ est un point qui	La composante λ^0 du point de départ est donnée par :	Afin d'obtenir (x^0, s^0), les composantes modifiées de $(\tilde{x}, \tilde{s})$ sont :	(x^0, λ^0, s^0) est un point
vérifie la contrainte de positivité : $(x, s) > 0$.	$\tilde{\lambda}$.	toutes les composantes.	strictement positif en x et s.

TAB. 6.2 – Points principaux de comparaison avec les méthodes de calcul du point de départ introduites en Programmation Linéaire.

Le tableau TAB. 6.2 ci-dessus clôture la description théorique de l'application de l'heuristique de M. Gertz, J. Nocedal et A. Sartenaer, à la programmation linéaire. Afin de tester son efficacité au départ de l'algorithme MPC, cette heuristique est implémentée à l'aide du logiciel Matlab, le programme

résultant étant joint en Annexe.

6.4 Résultats numériques

Comme nous l'avions annoncé précédemment, l'heuristique de M. Gertz et ses confrères est appliquée au départ de l'algorithme MPC, dans le but de résoudre les problèmes tests décrits par la table TAB. 3.2, présentée au troisième chapitre. Les caractéristiques des points de départ, engendrés pour chaque problème, sont alors analysées ainsi que le nombre d'itérations nécessaire à l'algorithme MPC pour résoudre ces problèmes. Les résultats obtenus à l'issue des différents tests sont commentés au cours de cette section.

Afin de calculer le point de départ (x^0, λ^0, s^0) de l'algorithme MPC pour chaque problème test, la stratégie de Gertz et ses confrères est utilisée en considérant tour à tour les quatre règles de calcul des composantes en x et s de ce point. Soulignons d'une part, que lorsque la première règle, par exemple, est utilisée l'approche est désignée par « R1 ». Une appellation similaire est utilisée dans le cas des autres règles. D'autre part, la séparation de l'ensemble des problèmes tests en deux parties, à savoir Gamme 1 et Gamme 2, est maintenue afin de rester cohérents avec la présentation des résultats faites au troisième chapitre. Cependant, au vu des résultats similaires observés pour les deux gammes de problèmes tests, nous avons choisi de superposer les résultats obtenus lors de l'analyse des caractéristiques des points de départ générés, ainsi que lors de la présentation des résultats généraux.

6.4.1 Analyse du point de départ

Tout d'abord, nous avons pu constater au cours des tests effectués que les points de départ calculés sont des points strictement positifs en leurs composantes x et s. De plus, nous avons remarqué que les composantes en x et s des points de départ générés sont identiques que nous utilisions la première règle de calcul ou la deuxième. Il semble que ce fait soit dû au choix du paramètre β_1 dont la valeur est élevée ($\beta_1 = 1000$). En effet, les composantes des points obtenus à partir des valeurs préliminaires en x et s, en effectuant un pas complet le long de la direction affine $\Delta^A v$, n'atteignent pas cette valeur. Par conséquent, ces composantes prennent respectivement la valeur attribuée à β_1.

Ensuite, nous avons analysé les caractéristiques des points de départ calculés pour chaque problème test afin de déterminer si ces points peuvent être considérés comme de « bons » points de départ ou non. Dès lors, nous avons observé la manière dont ces points satisfaisaient les conditions recquises par

S.J. Wright et introduites au Chapitre 2. Dans un premier temps, la valeur de l'expression (2.1), représentant le caractère de non-admissibilité associé aux points de départ, est évaluée pour les différents points engendrés. De cette manière, la table TAB. 6.3 présente les différentes valeurs obtenues, pour les problèmes appartenant à la Gamme 1 et à la Gamme 2.

Problème	$\frac{\|(r_b^0, r_c^0)\|_2}{\mu_0}$			
	R1	R2	R3	R4
Gamme 1				
1	0.17	0.17	0.17	0.17
2	0.25	0.25	0.25	0.25
3	0.18	0.18	0.18	0.18
4	0.05	0.05	0.05	0.05
5	0.19	0.19	0.19	0.19
6	0.41	0.41	0.41	0.41
7	0.09	0.09	0.09	0.09
8	0.30	0.30	0.30	0.30
9	0.16	0.16	0.16	0.16
10	0.48	0.48	0.48	0.48
11	0.61	0.61	0.61	0.61
12	0.85	0.85	0.85	0.85
13	1.20	1.20	1.20	1.20
Gamme 2				
14	0.11	0.11	0.11	0.11
15	0.29	0.29	0.29	0.29
16	1.01	1.01	1.01	1.01
17	1.83	1.83	1.83	1.83
18	4.61	4.61	4.61	4.61
19	2.82	2.82	2.82	2.82
20	0.83	0.83	0.83	0.83
21	1.46	1.46	1.46	1.46
22	0.79	0.79	0.79	0.79
23	1.21	1.21	1.21	1.21
24	1.83	1.83	1.83	1.83
25	3.41	3.41	3.41	3.41

TAB. 6.3 – Caractère de non-admissibilité du point de départ de l'algorithme MPC pour la résolution des problèmes de la Gamme 1 et de la Gamme 2.

Il suit de ces résultats que la valeur représentant le caractère de non-admissibilité, associée aux points de départ générés, est similaire quelle que soit la règle de calul de x^0 et s^0 utilisée. De plus, cette valeur est petite, ce qui signifie que ces points ne violent pas trop les contraintes d'égalité des problèmes primal et dual associés.

Dans un deuxième temps, nous avons évalué le caractère de centralité des points de départ obtenus, en calculant la mesure de proximité relative. La table TAB. 6.4 reprend ainsi la valeur associée à chaque point de départ engendré pour les problèmes de la Gamme 1 et de la Gamme 2.

Problème	$\frac{\|X^0S^0e-\mu_0e\|_2}{\mu_0}$			
	R1	R2	R3	R4
Gamme 1				
1	0	0	$2.17\ 10^{-4}$	$1.77\ 10^{-7}$
2	0	0	$3.81\ 10^{-4}$	$3.22\ 10^{-7}$
3	0	0	$7.65\ 10^{-4}$	$6.38\ 10^{-7}$
4	0	0	$4.72\ 10^{-4}$	$4.17\ 10^{-7}$
5	0	0	$5.66\ 10^{-4}$	$5.36\ 10^{-7}$
6	0	0	$5.49\ 10^{-4}$	$4.40\ 10^{-7}$
7	0	0	$3.34\ 10^{-4}$	$2.85\ 10^{-7}$
8	0	0	$5.46\ 10^{-4}$	$4.79\ 10^{-7}$
9	0	0	$1.48\ 10^{-4}$	$1.53\ 10^{-7}$
10	0	0	$4.08\ 10^{-4}$	$3.78\ 10^{-7}$
11	0	0	$3.50\ 10^{-4}$	$3.01\ 10^{-7}$
12	0	0	$2.26\ 10^{-4}$	$1.91\ 10^{-7}$
13	0	0	$3.77\ 10^{-4}$	$3.16\ 10^{-7}$
Gamme 2				
14	0	0	$4.35\ 10^{-4}$	$3.81\ 10^{-7}$
15	0	0	$5.32\ 10^{-4}$	$4.35\ 10^{-7}$
16	0	0	$3.04\ 10^{-4}$	$2.54\ 10^{-7}$
17	0	0	$4.95\ 10^{-4}$	$4.36\ 10^{-7}$
18	0	0	$4.87\ 10^{-4}$	$4.25\ 10^{-7}$
19	0	0	$4.35\ 10^{-4}$	$3.99\ 10^{-7}$
20	0	0	$8.10\ 10^{-4}$	$6.98\ 10^{-7}$
21	0	0	$4.15\ 10^{-4}$	$3.65\ 10^{-7}$
22	0	0	$2.74\ 10^{-4}$	$2.51\ 10^{-7}$
23	0	0	$3.37\ 10^{-4}$	$2.86\ 10^{-7}$
24	0	0	$6.85\ 10^{-4}$	$5.89\ 10^{-7}$
25	0	0	$4.55\ 10^{-4}$	$3.85\ 10^{-7}$

TAB. 6.4 – Mesure de proximité du point de départ de l'algorithme MPC pour la résolution des problèmes de la Gamme 1 et de la Gamme 2.

Pour l'ensemble des problèmes tests, nous remarquons que les points engendrés par l'approche de Gertz et ses confrères sont des points proches de la trajectoire centrale $\mathcal{C}$, voire sur $\mathcal{C}$ lors de l'utilisation des deux premières règles de calcul. En effet, dans ce cas, nous constatons que toutes les composantes de x^0 et s^0 sont identiques. Cette observation semble provenir du choix de la valeur attribuée au paramètre β_1.

Il semble donc que les points de départ générés en utilisant l'approche de M. Gertz et ses confrères puissent être considérer comme de « bons » points de départ, puisqu'ils satisfont relativement bien les conditions suggérées par S.J. Wright.

6.4.2 Résultats généraux

Avant de comparer les résultats des tables TAB. 6.3 et TAB. 6.4 avec ceux présentés au troisième chapitre, le nombre d'itérations nécessaire à l'algorithme MPC, pour résoudre les problèmes tests, est repris dans la table TAB. 6.5.

Problème	♯ d'itérations			
	R1	R2	R3	R4
Gamme 1				
1	9	9	9	9
2	10	10	10	10
3	10	10	10	10
4	11	11	11	11
5	10	10	10	10
6	12	12	12	12
7	9	9	9	9
8	11	11	11	11
9	8	8	8	8
10	12	12	12	12
11	12	12	12	12
12	12	12	12	12
13	14	14	14	14
Gamme 2				
14	10	10	10	10
15	10	10	10	10
16	13	13	13	13
17	12	12	12	12
18	13	13	13	13
19	13	13	13	13
20	12	12	12	12
21	12	12	12	12
22	11	11	11	11
23	11	11	11	11
24	12	12	12	12
25	13	13	13	13

TAB. 6.5 – Nombre d'itérations nécessaire pour résoudre les problèmes de la Gamme 1 et de la Gamme 2.

Nous observons que le nombre d'itérations nécessaire à l'algorithme MPC est identique quelle que soit la règle de calcul utilisée pour déterminer les composantes en x et s du point de départ. Cela peut s'expliquer par le fait que les points de départ générés pour un même problème sont très proches, voire les mêmes, quelle que soit la règle de calcul sélectionnée.

6.4.3 Comparaison avec les résultats établis au Chapitre 3

Comparons à présent les points de départ obtenus en suivant l'approche de Gertz et ses confrères avec ceux calculés par les méthodes de Mehrotra, Gondzio et ses confrères, et Zhang, en analysant leurs caractères de centralité et de non-admissibilité. Au vu des valeurs reprises dans les tables TAB. 3.3 et TAB 3.5, l'heuristique décrite au cours de ce chapitre permet de calculer des points de départ plus réalisables que ceux générés par les trois autres approches, pour les problèmes correspondants. En effet, la valeur de l'expression (2.1), relative à ces points de départ est meilleure [TAB. 6.3]. De plus, à partir des tables TAB. 3.4 et TAB. 3.6, nous voyons que les points de départ engendrés par la méthode de Gertz et ses confrères satisfont mieux la condition de proximité dans le sens où la mesure de proximité qui leur est relative est plus petite [TAB. 6.4].

Dès lors, il semble que les points engendrés par cette heuristique soient de « meilleurs » points de départ. Nous sommes donc en droit de nous attendre à ce que l'algorithme MPC résolve plus rapidement les problèmes correspondants, au départ de tels points. Cependant, si nous comparons le nombre d'itérations nécessaire à celui-ci pour résoudre les problèmes tests, nous observons que ce n'est pas le cas. En effet, en reprenant les tables TAB. 3.7 et TAB. 3.8, il suit que l'algorithme MPC, au départ d'un point déterminé par l'approche de Gertz et ses confrères, nécessite plus d'itérations pour la résolution des problèmes de la Gamme 1, comme de la Gamme 2 [TAB. 6.5], voire le même nombre d'itérations qu'à partir d'un point généré par l'approche de Zhang.

Une explication possible de la différence de performance de l'algorithme MPC, constatée au paragraphe précédent, pourrait venir de la proximité du point de départ par rapport à la solution optimale. Dès lors, nous avons analysé les distances séparant les composantes en x, resp. s, des points de départ engendrés par les différentes approches, aux composantes correspondantes de la solution. Les tables TAB. 6.6 et TAB. 6.7, ainsi que les tables TAB. 6.8 et TAB. 6.9, donnent respectivement les résultats obtenus pour les problèmes de la Gamme 1 et de la Gamme 2, résultats obtenus en évaluant les distances $||x^* - x^0||$ et $||s^* - s^0||$ à l'aide de la norme-2.

Gamme 1			
Problème	$\|\|x^* - x^0\|\|_2$		
	Mehrotra	Gondzio	Zhang
1	$8.07\ 10^1$	$1.78\ 10^1$	$3.38\ 10^1$
2	$4.04\ 10^1$	$2.61\ 10^1$	$6.04\ 10^1$
3	$1.45\ 10^2$	$7.48\ 10^1$	$1.69\ 10^2$
4	$6.43\ 10^1$	$5.58\ 10^1$	$8.63\ 10^1$
5	$1.51\ 10^2$	$7.55\ 10^1$	$1.98\ 10^2$
6	$2.43\ 10^2$	$8.11\ 10^1$	$2.16\ 10^2$
7	$1.18\ 10^2$	$3.10\ 10^1$	$1.61\ 10^2$
8	$4.92\ 10^1$	$2.72\ 10^1$	$5.66\ 10^1$
9	$5.88\ 10^1$	8.05	$5.94\ 10^1$
10	$6.18\ 10^1$	$4.67\ 10^1$	$9.34\ 10^1$
11	$1.05\ 10^2$	$6.87\ 10^1$	$1.49\ 10^2$
12	$3.35\ 10^2$	$8.30\ 10^1$	$3.08\ 10^2$
13	$5.96\ 10^2$	$1.87\ 10^2$	$6.22\ 10^2$
	$\|\|s^* - s^0\|\|_2$		
	Mehrotra	Gondzio	Zhang
1	3.03	6.94	6.94
2	8.22	7.75	8.21
3	6.63	6.58	6.61
4	2.50	4.59	4.59
5	9.57	6.58	6.58
6	9.37	8.79	9.06
7	5.50	5.47	5.47
8	5.98	8.20	8.30
9	4.01	6.72	6.75
10	$1.05\ 10^1$	9.43	9.47
11	$1.14\ 10^1$	$1.04\ 10^1$	$1.05\ 10^1$
12	7.64	$1.16\ 10^1$	$1.20\ 10^1$
13	$1.86\ 10^1$	$1.32\ 10^1$	$1.34\ 10^1$

TAB. 6.6 – Mesure de la distance séparant les points x^0, resp s^0, déterminés par les approches de Mehrotra, Gondzio et ses confrères, et Zhang, et les points x^*, resp. s^*, pour les problèmes de la Gamme 1.

Gamme 1				
Problème	$\|\|x^* - x^0\|\|_2$			
	R1	R2	R3	R4
1	$7.03\ 10^3$	$7.03\ 10^3$	$7.03\ 10^3$	$7.03\ 10^3$
2	$8.34\ 10^3$	$8.34\ 10^3$	$8.35\ 10^3$	$8.35\ 10^3$
3	$7.67\ 10^3$	$7.67\ 10^3$	$7.68\ 10^3$	$7.68\ 10^3$
4	$4.96\ 10^3$	$4.96\ 10^3$	$4.96\ 10^3$	$4.96\ 10^3$
5	$7.99\ 10^3$	$7.99\ 10^3$	$8.00\ 10^3$	$8.00\ 10^3$
6	$9.90\ 10^3$	$9.90\ 10^3$	$9.91\ 10^3$	$9.92\ 10^3$
7	$5.79\ 10^3$	$5.79\ 10^3$	$5.79\ 10^3$	$5.80\ 10^3$
8	$8.91\ 10^3$	$8.91\ 10^3$	$8.92\ 10^3$	$8.92\ 10^3$
9	$6.87\ 10^3$	$6.87\ 10^3$	$6.87\ 10^3$	$6.87\ 10^3$
10	$1.04\ 10^4$	$1.04\ 10^4$	$1.04\ 10^4$	$1.04\ 10^4$
11	$1.11\ 10^4$	$1.11\ 10^4$	$1.11\ 10^4$	$1.11\ 10^4$
12	$1.22\ 10^4$	$1.22\ 10^4$	$1.22\ 10^4$	$1.22\ 10^4$
13	$1.39\ 10^4$	$1.39\ 10^4$	$1.39\ 10^4$	$1.39\ 10^4$
	$\|\|s^* - s^0\|\|_2$			
	R1	R2	R3	R4
1	$7.07\ 10^3$	$7.07\ 10^3$	$7.07\ 10^3$	$7.07\ 10^3$
2	$8.36\ 10^3$	$8.36\ 10^3$	$8.36\ 10^3$	$8.37\ 10^3$
3	$7.74\ 10^3$	$7.74\ 10^3$	$7.74\ 10^3$	$7.75\ 10^3$
4	$5.00\ 10^3$	$5.00\ 10^3$	$5.00\ 10^3$	$5.00\ 10^3$
5	$8.06\ 10^3$	$8.06\ 10^3$	$8.06\ 10^3$	$8.06\ 10^3$
6	$1.00\ 10^4$	$1.00\ 10^4$	$1.00\ 10^4$	$1.00\ 10^4$
7	$5.83\ 10^3$	$5.83\ 10^3$	$5.83\ 10^3$	$5.83\ 10^3$
8	$8.94\ 10^3$	$8.94\ 10^3$	$8.94\ 10^3$	$8.95\ 10^3$
9	$6.93\ 10^3$	$6.93\ 10^3$	$6.93\ 10^3$	$6.93\ 10^3$
10	$1.05\ 10^4$	$1.05\ 10^4$	$1.05\ 10^4$	$1.05\ 10^4$
11	$1.11\ 10^4$	$1.11\ 10^4$	$1.11\ 10^4$	$1.11\ 10^4$
12	$1.24\ 10^4$	$1.24\ 10^4$	$1.24\ 10^4$	$1.24\ 10^4$
13	$1.41\ 10^4$	$1.41\ 10^4$	$1.41\ 10^4$	$1.41\ 10^4$

TAB. 6.7 – Mesure de la distance séparant les points x^0, resp s^0, déterminés par les approches R1, R2, R3, et R4, et les points x^*, resp. s^*, pour les problèmes de la Gamme 1.

Gamme 2			
Problème	$\|\|x^* - x^0\|\|_2$		
	Mehrotra	Gondzio	Zhang
14	5.47	6.27	7.05
15	5.01	6.57	8.89
16	7.67	8.53	$1.13\ 10^1$
17	$1.13\ 10^1$	$1.37\ 10^1$	$2.91\ 10^1$
18	$1.54\ 10^1$	$1.62\ 10^1$	$3.32\ 10^1$
19	$1.74\ 10^1$	$1.35\ 10^1$	$2.86\ 10^1$
20	$1.14\ 10^1$	$1.33\ 10^1$	$2.16\ 10^1$
21	9.60	$1.13\ 10^1$	$1.90\ 10^1$
22	6.92	7.77	$1.08\ 10^1$
23	6.69	8.93	$1.14\ 10^1$
24	$1.44\ 10^1$	$1.55\ 10^1$	$3.67\ 10^1$
25	$1.28\ 10^1$	$1.41\ 10^1$	$2.60\ 10^1$
	$\|\|s^* - s^0\|\|_2$		
	Mehrotra	Gondzio	Zhang
14	6.07	5.68	5.68
15	8.60	7.95	8.18
16	9.16	$1.25\ 10^1$	$1.32\ 10^1$
17	$1.81\ 10^1$	$1.49\ 10^1$	$1.56\ 10^1$
18	$1.75\ 10^1$	$2.04\ 10^1$	$2.13\ 10^1$
19	$1.79\ 10^1$	$1.72\ 10^1$	$1.79\ 10^1$
20	$1.63\ 10^1$	$1.10\ 10^1$	$1.13\ 10^1$
21	$1.64\ 10^1$	$1.37\ 10^1$	$1.38\ 10^1$
22	7.63	$1.13\ 10^1$	$1.13\ 10^1$
23	$1.12\ 10^1$	$1.31\ 10^1$	$1.35\ 10^1$
24	$1.59\ 10^1$	$1.45\ 10^1$	$1.51\ 10^1$
25	$1.84\ 10^1$	$1.82\ 10^1$	$1.93\ 10^1$

TAB. 6.8 – Mesure de la distance séparant les points x^0, resp s^0, déterminés par les approches de Mehrotra, Gondzio et ses confrères, et Zhang, et les points x^*, resp. s^*, pour les problèmes de la Gamme 2.

Gamme 2				
Problème	$\|\|x^* - x^0\|\|_2$			
	R1	R2	R3	R4
14	$6.63\ 10^3$	$6.63\ 10^3$	$6.64\ 10^3$	$6.64\ 10^3$
15	$8.88\ 10^3$	$8.88\ 10^3$	$8.89\ 10^3$	$8.86\ 10^3$
16	$1.31\ 10^4$	$1.31\ 10^4$	$1.31\ 10^4$	$1.31\ 10^4$
17	$1.68\ 10^4$	$1.68\ 10^4$	$1.69\ 10^4$	$1.69\ 10^4$
18	$2.23\ 10^4$	$2.23\ 10^4$	$2.24\ 10^4$	$2.24\ 10^4$
19	$1.89\ 10^4$	$1.89\ 10^4$	$1.89\ 10^4$	$1.89\ 10^4$
20	$1.37\ 10^4$	$1.37\ 10^4$	$1.38\ 10^4$	$1.38\ 10^4$
21	$1.52\ 10^4$	$1.52\ 10^4$	$1.52\ 10^4$	$1.52\ 10^4$
22	$1.20\ 10^4$	$1.20\ 10^4$	$1.20\ 10^4$	$1.20\ 10^4$
23	$1.39\ 10^4$	$1.39\ 10^4$	$1.39\ 10^4$	$1.39\ 10^4$
24	$1.73\ 10^4$	$1.73\ 10^4$	$1.73\ 10^4$	$1.73\ 10^4$
25	$2.00\ 10^4$	$2.00\ 10^4$	$2.00\ 10^4$	$2.00\ 10^4$
	$\|\|s^* - s^0\|\|_2$			
	R1	R2	R3	R4
14	$6.31\ 10^3$	$6.63\ 10^3$	$6.63\ 10^3$	$6.63\ 10^3$
15	$8.89\ 10^3$	$8.89\ 10^3$	$8.89\ 10^3$	$8.89\ 10^3$
16	$1.31\ 10^4$	$1.31\ 10^4$	$1.31\ 10^4$	$1.31\ 10^4$
17	$1.68\ 10^4$	$1.68\ 10^4$	$1.68\ 10^4$	$1.69\ 10^4$
18	$2.24\ 10^4$	$2.24\ 10^4$	$2.24\ 10^4$	$2.24\ 10^4$
19	$1.89\ 10^4$	$1.89\ 10^4$	$1.89\ 10^4$	$1.89\ 10^4$
20	$1.37\ 10^4$	$1.37\ 10^4$	$1.37\ 10^4$	$1.37\ 10^4$
21	$1.52\ 10^4$	$1.52\ 10^4$	$1.52\ 10^4$	$1.52\ 10^4$
22	$1.20\ 10^4$	$1.20\ 10^4$	$1.20\ 10^4$	$1.20\ 10^4$
23	$1.39\ 10^4$	$1.39\ 10^4$	$1.39\ 10^4$	$1.39\ 10^4$
24	$1.73\ 10^4$	$1.73\ 10^4$	$1.73\ 10^4$	$1.73\ 10^4$
25	$2.00\ 10^4$	$2.00\ 10^4$	$2.00\ 10^4$	$2.00\ 10^4$

TAB. 6.9 – Mesure de la distance séparant les points x^0, resp s^0, déterminés par les approches R1, R2, R3, et R4, et les points x^*, resp. s^*, pour les problèmes de la Gamme 2.

Au vu des résultats obtenus, nous observons en effet que les points générés en suivant l'approche de Gertz et ses confrères, [TAB. 6.7 et TAB. 6.9], s'avèrent être beaucoup plus éloignés de la solution que les points déterminés à partir de la méthode de Mehrotra, de Gondzio et ses confrères, ou encore de Zhang [TAB. 6.6 et TAB. 6.8].

6.4.4 Conclusion

Par conséquent, malgré le fait que les points de départ engendrés par l'approche de Gertz et ses confrères semblent « meilleurs », au sens où l'entend S.J. Wright, que les points générés par les méthodes de Mehrotra, de Gondzio et ses confrères, ainsi que de Zhang, l'algorithme MPC ne résoud pas plus rapidement l'ensemble des problèmes tests, au départ de ceux-ci. Cette observation semble être justifiée par le fait que ces points sont plus éloignés de la solution que les points générés par les trois autres approches. Par conséquent, un point de départ de l'algorithme MPC peut se trouver près de la trajectoire centrale, voire sur celle-ci, et satisfaire relativement bien les conditions d'égalité du problème à résoudre, sans pour autant être proche de la solution optimale. Dans ce cas, l'algorithme sélectionné nécessite plus de temps, en nombre d'itérations, pour résoudre le problème donné.

Conclusion

Au cours de ce mémoire, nous avons analysé et comparé quatre méthodes de calcul de points de départ pour des algorithmes de points intérieurs de type primal-dual, d'un point de vue théorique comme pratique. Pour cela, ces méthodes ont été appliquées au départ de l'algorithme primal-dual de points intérieurs de Mehrotra, dit algorithme MPC, afin de résoudre un ensemble de problèmes tests. De cette manière, nous avons pu constater que les conditions théoriques recquises, selon S.J. Wright, pour qu'un point soit considéré comme un « bon » point de départ, ne sont pas toujours satisfaites par les points engendrés à partir des méthodes présentées. De plus, nous avons observé que la résolution des problèmes tests ne s'avérait pas plus rapide au départ de « bon » points de départ, du moins si l'on considére d'autres caractérisques associées aux points de départ telle que la distance de ces points à la solution optimale.

Soulignons qu'à travers les différents tests que nous avons effectués, nous avons attribué aux paramères utilisés dans les méthodes de calcul de points de départ de Gondzio et ses confrères, de Zhang, et de Gertz et ses confrères, les valeurs suggérées par ces mathématiciens. Toutefois, il pourrait s'avérer intéressant de les modifier dans le but d'analyser les caractéristiques des points de départ obtenus de cette manière ainsi que l'impact que cela pourrait avoir sur la résolution d'un ensemble de problèmes tests. Malheureusement, étant limités dans le temps, il ne nous a pas été possible d'effectuer ces tests supplémentaires.

D'autre part, l'application de l'heuristique de M. Gertz et ses confrères au départ d'un algorithme primal-dual de points intérieurs en programmation non linéaire s'avère également être un prolongement aux développements effectuées au cours de ce mémoire. Ce travail est d'ailleurs en cours de développement, par les mathématiciens M. Gertz, J. Nocedal et A. Sartenaer.

Troisième partie

Annexe

Annexe A

Programmes

A.1 Création des problèmes tests

```
format long

% Choix du nombre de variables et du nombre de contraintes du problème :

n=input('entrez une valeur pour n : ');
m=input('entrez une valeur pour m : ');

% Création du vecteur c, dans R^n, à l'aide de la fonction random :

c=rand(n,1);
c=c;
save res31.mat c -ASCII

% Création de la matrice A, m x n, de rang m, à l'aide de la fonction
% random :

A=rand(m,n);
A=A;
save res32.mat A -ASCII

% Choix, par l'utilisateur, des coefficients l > 0 de la combinaison
% linéaire (3.3) :

for j=1:n
    l(j)=input('entrez une valeur > 0 pour les coef de la combili : ');
    while l(j) <= 0
        l(j)=input('une valeur > 0 ! : ');
    end
```

```
end

% Calcul du vecteur b, combinaison linéaire des colonnes de A et donné
% par (3.3) :

for i=1:m
    interm(i)=0;
end

for j=1:n
    for i=1:m
        b(i)=interm(i)+l(j)*A(i,j);
        interm(i)=b(i);
    end
end
b=b';
save res33.mat b -ASCII
format long

% Choix du nombre de variables et du nombre de contraintes du problème :

n=input('entrez une valeur pour n : ');
m=input('entrez une valeur pour m : ');

% Création du vecteur c, dans R^n, à l'aide de la fonction random :

c=rand(n,1);
c=c;
save res31.mat c -ASCII

% Création de la matrice A, m x n, de rang m, à l'aide de la fonction random :

A=rand(m,n);
A=A;
save res32.mat A -ASCII

% Choix des coefficients l > 0 de la combinaison linéaire (3.3), à l'aide de
% la fonction random :

for j=1:n
    l(j)=rand(1)
end

% Calcul du vecteur b, combinaison linéaire des colonnes de A et donné
```

```
% par (3.3) :

for i=1:m
    interm(i)=0;
end

for j=1:n
    for i=1:m
        b(i)=interm(i)+l(j)*A(i,j);
        interm(i)=b(i);
    end
end
b=b';
save res33.mat b -ASCII
```

A.2 Programme principal

```
% Auteur                  : Braibant Anne-Sophie
% Dernières mises à jour : 09 mai 2003
% Contenu                 : Algorithme Predicteur-Correcteur de Mehrotra(MPC)

% L'algorithme MPC implémente une méthode de points intérieurs permettant
% de résoudre des problèmes de type primal de la forme :
%                        min c^T x
%                        s.c. : Ax  = b
%                                x >= 0,
% où c,x \in R^n, b \in R^m et A \in R^(m x n) de rang m.

format long

% Initialisation des données, c, b et A, du problème à résoudre,
% du nombre de variables n, du nombre de contraintes m, ainsi que
% des vecteurs unités u et e, resp. dans R^n et R^m :

global c
global A
global b
global regle

load res31.mat -ASCII
c=res31;

load res32.mat -ASCII
A=res32;
```

```
load res33.mat -ASCII
b=res33;

n=length(c);
m=length(b);
u=ones(n,1);
e=ones(m,1);

% Choix d'une méthode de calcul du point de départ (x,l,s) :

disp ('Nom des méthodes de calcul du point de départ : ')
disp ('1. mehrotra')
disp ('2. gondzio')
disp ('3. zhang')
disp ('4. gertz')
rep = input('Entrez le numéro de votre choix : ');

% Rem : pour la méthode de Gertz et ses confrères, les quatre règles de
%       calcul de x^0 et s^0 sont testées.

if ( rep == 4 )
    disp ('Choix de la règle pour calculer les composantes x et s du
           point de départ : ')
    disp ('Règle 1.')
    disp ('Règle 2.')
    disp ('Règle 3.')
    disp ('Règle 4.')
    regle = input('Entrez le numéro de votre choix : ');
end

if ( rep == 1 )
    [x,l,s] = mehrotra;
elseif ( rep == 2 )
    [x,l,s] = gondzio;
elseif ( rep == 3 )
    [x,l,s] = zhang;
elseif rep == 4
    [x,l,s] = gertz;
end

% Calcul des valeurs initiales des vecteurs résidus r_b et r_c, en norme-2,
% de la mesure de dualité \mu et du caractère de non-admissibilité,
% donné par la relation (2.1) et noté ici par infaisab :
```

```
r_b=A*x-b;
infp=norm(r_b);
r_c=A'*l+s-c;
infd=norm(r_c);
mu=(x'*s)/n;
tot=[r_b ; r_c];
infaisab=(norm(tot))/mu;

% Calcul de la mesure de proximité associée au point de départ, définie par
% (1.23) et appelée prox :

X0=diag(x);
S0=diag(s);
diff=X0*S0*u-mu*u;
norme=norm(diff);
prox=norme/mu;

% Début algorithme :

iter=0;
itermax=n;
alonzy=1;
epsilon=1.0e-5;

while ((iter <= itermax) & alonzy)

    iter=iter+1;

    % Calcul de la direction affine obtenue par résolution du système (1.41) :

    S=diag(s);
    X=diag(x);
    L=diag(l);
    r_xs=X*S*u;
    G=[zeros(n,n) A' eye(n,n) ; A zeros(m,m) zeros(m,n) ; S zeros(n,m) X];
    h=[-r_c ; -r_b ; -r_xs];
    d_af=G\h;
    delta_xaf=d_af(1:n);
    delta_laf=d_af(n+1:n+m);
    delta_saf=d_af(n+m+1:m+2*n);

    % Calcul du pas primal dans la direction affine :

    negxaf=find(delta_xaf<0);
    xtemp=x(negxaf);
```

```
dxtemp=-delta_xaf(negxaf);
test=xtemp./dxtemp;
valx=[ 1 ; min(xtemp./dxtemp)];
if length(negxaf)~=0
    alpha_priaf=min(valx);
else
    alpha_priaf=1;
end

% Calcul du pas dual dans la direction affine :

delta_saf=delta_saf;
negsaf=find(delta_saf<0);
stemp=s(negsaf);
dstemp=-delta_saf(negsaf);
testbis=stemp./dstemp;
vals=[ 1 ; min(stemp./dstemp)];
if length(negsaf)~=0
    alpha_duaf=min(vals);
else
    alpha_duaf=1;
end

% Calcul de la valeur hypothétique de mu, donnée par (1.43) :

mu_af=(x+alpha_priaf*delta_xaf)'*(s+alpha_duaf*delta_saf)/n;

% Calcul du paramètre de centrage, défini en (1.44) :

sigma=(mu_af/mu)^3;

% Calcul combiné des directions de centrage et correctrice, par résolution
% du système (1.42) :

delta_Xaf=diag(delta_xaf);
delta_Saf=diag(delta_saf);
r_xs_bis=-sigma*mu*u+delta_Xaf*delta_Saf*u;
h=[zeros(n,1) ; zeros(m,1) ; -r_xs_bis];
d_cc=G\h;
delta_xcc=d_cc(1:n);
delta_lcc=d_cc(n+1:n+m);
delta_scc=d_cc(n+m+1:2*n+m);

% Calcul de la direction totale, combinaison des deux directions
% calculées précédemment :
```

```
d_x=delta_xaf+delta_xcc;
d_l=delta_laf+delta_lcc;
d_s=delta_saf+delta_scc;

% Calcul du pas primal maximal jusqu'à la frontière de l'orthant de
% non-négativité, défini par (x,s) >= 0 :

negdx=find(d_x<0);
ptemp=x(negdx);
dptemp=-d_x(negdx);
pasx=[ 1 ; min(ptemp./dptemp) ];
if length(negdx)~=0
    alpha_primax=min(pasx);
else
    alpha_primax=1;
end

% Calcul du pas dual maximal jusqu'à la frontière de l'orthant de
% non-négativité, défini par (x,s) >= 0 :

negds=find(d_s<0);
dtemp=s(negds);
ddtemp=-d_s(negds);
pass=[ 1 ; min(dtemp./ddtemp) ];
if length(negds)~=0
    alpha_dumax=min(pass);
else
    alpha_dumax=1;
end

% Calcul des pas primal et dual effectués le long de la direction totale :

alpha=[ 0.995*alpha_primax ; 1 ];
alpha_pri=min(alpha);
alpha=[ 0.995*alpha_dumax ; 1];
alpha_du=min(alpha);

% Calcul du nouvel itéré :

x=x+alpha_pri*d_x;
l=l+alpha_du*d_l;
s=s+alpha_du*d_s;

% Mise à jour des vecteurs résidus, de la mesure de dualité et de la
```

```
    % condition d'arret :

    r_b=A*x-b;
    infp=norm(r_b);
    r_c=A'*l+s-c;
    infd=norm(r_c);
    mu=(x'*s)/n;

    alonzy=(infp>epsilon)|(infd>epsilon)|(mu>epsilon);

end
```

A.3 Méthode de Mehrotra

```
function[x,l,s]=mehrotra(c,A,b)

format long

global c
global A
global b

n=length(c);
m=length(b);
u=ones(n,1);

% Calcul du point préliminaire (x,l,s), solution de la paire
% de problèmes (2.4)-(2.5) :

x=A'*inv(A*A')*b;
l=inv(A*A')*A*c;
s=c-A'*l;

% Calcul des quatre scalaires, donnés par (2.7)-(2.10) :

q=[ -1.5*min(x) ; 0 ];
deltax=max(q);
qbis=[ -1.5*min(s) ; 0 ];
deltas=max(qbis);
xb=x+deltax*u;
sb=s+deltas*u;
prod=xb'*sb;
div=prod/sum(sb);
term2=0.5*div;
```

```
tildex=deltax+term2;
test=sum(xb);
div=prod/sum(xb);
term2=0.5*div;
tildes=deltas+term2;
phi_x(1:n,1)=tildex;
phi_s(1:n,1)=tildes;

% Mise à jour de x et s afin d'obtenir le point de départ :

x=x+phi_x(1:n,1);
s=s+phi_s(1:n,1);
```

A.4 Méthode de Gondzio et ses confrères

```
function[x,l,s]=gondzio(c,A,b)

format long

global c
global A
global b

n=length(c);
m=length(b);
u=ones(n,1);

% Calcul du point préliminaire (x,l,s), solution de la paire
% de problèmes (2.26)-(2.27), en résolvant le système des conditions
% d'optimalité associé à chacun de ces problèmes :

K1=[eye(n,n) A' ; A zeros(m,m)];
h1=[-c ; b];
d1=K1\h1;
x=d1(1:n);

K2=[eye(m,m) zeros(m,n) A eye(m,m) ;
    zeros(n,m) eye(n,n) eye(n,n) zeros(n,m) ;
    A' eye(n,n) zeros(n,n) zeros(n,m) ;
    eye(m,m) zeros(m,n) zeros(m,n) zeros(m,m)];
h2=[-b ; zeros(n,1) ; c ; zeros(m,1)];
d2=K2\h2;
l=d2(1:m);
s=d2(m+1:m+n);
```

```
% Mise à jour de x et s pour obtenir le point de départ :

delta=1;

for i=1:n
    x(i) = max( x(i), delta );
    s(i) = max( s(i), delta );
end
```

A.5 Méthode de Zhang

```
function[x,l,s]=zhang(c,A,b)

global c
global A
global b

n=length(c);
m=length(b);

% Calcul la matrice B :

[Q,R]=qr(A');
Q1=Q(1:n,1:m);
Q2=Q(1:n,m+1:n);
B=Q2';

% Calcul du point préliminaire (x,l,s) :

x=A\b;

h=B*c;
s=B\h;

k=c-s;
l=A'\k;

% Mise à jour de x et s pour obtenir le point de départ :

zeta=1;
for i=1:n
    x(i) = max( x(i), zeta );
    s(i) = max( s(i), zeta );
```

```
end
```

A.6 Méthode de Gertz et ses confrères

```
function[x,l,s]=gertz(c,A,b)

format long

global A
global b
global c
global regle

n=length(c);
m=length(b);
u=ones(n,1);

% Initialisation des paramètres utilisés par l'approche de Gertz et
% ses confrères :

delta=1;
beta_1=1000;
beta_2=2;

% Initialisation des valeurs préliminaires de x et s :

tilde_x=delta*u;
tilde_s=delta*u;

% Calcul de la direction affine par résolution de système (1.15) :

TILDE_X=diag(tilde_x);
TILDE_S=diag(tilde_s);

G=[zeros(n,n) A' eye(n,n) ;
   A zeros(m,m) zeros(m,n) ;
   TILDE_S zeros(n,m) TILDE_X];
h=[zeros(n,1) ; zeros(m,1) ; -TILDE_X*TILDE_S*u];
delta_va=G\h;
delta_xa=delta_va(1:n);
delta_sa=delta_va(n+m+1:2*n+m);

% Calcul du point obtenu à partir des valeurs préliminaires de x et s,
% en effectuant un pas total le long de la direction affine :
```

```
newtilde_x=tilde_x+delta_xa;
newtilde_s=tilde_s+delta_sa;

% Calcul des vecteurs mesurant la violation des contraintes de positivité
% primales et duales, définis par les relations (6.2) et (6.3) :

vio_pri=zeros(n,1);
vio_dual=zeros(n,1);
for i=1:n
    vio_pri(i)=max(0,-newtilde_x(i));
    vio_dual(i)=max(0,-newtilde_s(i));
end

% Calcul du scalaire représentant la violation des contraintes de positivité
% maximale, donné par (6.4) :

vio_total=[vio_pri ; vio_dual];
xi=max(vio_total);

% Calcul du multiplicateur de Lagrange l :

l=inv(A*A')*A*c;

% Calcul des composantes x et s du point de départ, à l'aide d'une des quatre
% règles proposées :

x=zeros(n,1);
s=zeros(n,1);

if regle == 1
    for i=1:n
        x(i)=max(beta_1,newtilde_x(i));
        s(i)=max(beta_1,newtilde_s(i));
    end
elseif regle == 2
    for i=1:n
        choix_x=[beta_1,newtilde_x(i),vio_pri(i)];
        x(i)=max(choix_x);
        choix_s=[beta_1,newtilde_s(i),vio_dual(i)];
        s(i)=max(choix_s);
    end
elseif regle == 3
    x=newtilde_x+beta_1*u+beta_2*vio_pri;
    s=newtilde_s+beta_1*u+beta_2*vio_dual;
```

```
elseif regle == 4
    x=newtilde_x+beta_1*u+(beta_2*xi)*u;
    s=newtilde_s+beta_1*u+(beta_2*xi)*u;
end
```

Bibliographie

[1] E.D. Andersen, J. Gondzio, C. Mészaráros, X. Xu, Implementation of Interior Point Methods for Large Scale Linear Programming, *Technical Report 1996.3*, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, Switzerland, 1996.

[2] R.H. Byrd, J.C. Gilbert, and J. Nocedal, A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming, *Mathematical Programming*, Vol. 89, pp. 149-185, 2000.

[3] R.H. Byrd, M.E. Hribar, and J. Nocedal, An interior point algorithm for large-scale nonlinear programming, *Siam J. Optimization*, Vol. 9, nř4, pp. 877-900, 1999.

[4] R.H. Byrd, G. Liu, J. Nocedal, On the local behavior of an interior point method for nonlinear programming, *Numerical Analysis*, D.F. Griffiths and D.J. Higham, Eds, Addison-Wesley Longman, Reading, MA, pp. 37-56, 1997.

[5] R. Fourer, D.M. Gay, B.W. Kernighan, AMPL : A Modeling Language For Mathematical Programming, *The Scientific Press*, 1993.

[6] M. Gertz, J. Nocedal, and A. Sartenaer, A starting-point strategy for nonlinear interior methods, en préparation.

[7] G.H. Golub, C.F. Van Loan, Matrix Conputations, *Johns Hopkins University Press*, 1996.

[8] http ://www-neos.mcs.anl.gov/

[9] S. Mehrotra, On the implementation of a primal-dual interior point method, *Siam J. Optimization*, Vol. 2, nř4, pp. 575-601, 1992.

[10] J. Nocedal, S.J. Wright, Numerical Optimization, *Springer Series in Operations Research*, 1999.

[11] J.J. Strodiot, Cours de Méthodes numériques d'optimisation, $2^{\text{ème}}$ licence, année 2002-2003.

[12] S.J. Wright, Primal-Dual Interior-Point Methods, *Siam*, 1997.

[13] Y. Zhang, On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem, *Siam J. Optimization*, Vol. 4, nř1, pp. 208-227, 1994.